

Guoliang Li · Jun Yang ·  
Joao Gama · Juggapong Natwichai ·  
Yongxin Tong (Eds.)

LNCS 11447

# Database Systems for Advanced Applications

24th International Conference, DASFAA 2019  
Chiang Mai, Thailand, April 22–25, 2019  
Proceedings, Part II

2  
Part II

 Springer

*Commenced Publication in 1973*

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

## Editorial Board Members

David Hutchison

*Lancaster University, Lancaster, UK*

Takeo Kanade

*Carnegie Mellon University, Pittsburgh, PA, USA*

Josef Kittler

*University of Surrey, Guildford, UK*

Jon M. Kleinberg

*Cornell University, Ithaca, NY, USA*

Friedemann Mattern

*ETH Zurich, Zurich, Switzerland*

John C. Mitchell

*Stanford University, Stanford, CA, USA*

Moni Naor

*Weizmann Institute of Science, Rehovot, Israel*

C. Pandu Rangan

*Indian Institute of Technology Madras, Chennai, India*

Bernhard Steffen

*TU Dortmund University, Dortmund, Germany*

Demetri Terzopoulos

*University of California, Los Angeles, CA, USA*

Doug Tygar

*University of California, Berkeley, CA, USA*



More information about this series at <http://www.springer.com/series/7409>


Guoliang Li · Jun Yang ·  
Joao Gama · Juggapong Natwichai ·  
Yongxin Tong (Eds.)

# Database Systems for Advanced Applications

24th International Conference, DASFAA 2019  
Chiang Mai, Thailand, April 22–25, 2019  
Proceedings, Part II

*Editors*

Guoliang Li  
Tsinghua University  
Beijing, China

Joao Gama   
University of Porto  
Porto, Portugal

Yongxin Tong  
Beihang University  
Beijing, China

Jun Yang  
Duke University  
Durham, NC, USA

Juggapong Natwichai  
Chiang Mai University  
Chiang Mai, Thailand

ISSN 0302-9743                      ISSN 1611-3349 (electronic)  
Lecture Notes in Computer Science  
ISBN 978-3-030-18578-7              ISBN 978-3-030-18579-4 (eBook)  
<https://doi.org/10.1007/978-3-030-18579-4>

LNCS Sublibrary: SL3 – Information Systems and Applications, incl. Internet/Web, and HCI

© Springer Nature Switzerland AG 2019, corrected publication 2019

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG  
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

# Preface

The International Conference on Database Systems for Advanced Applications DASFAA provides a leading international forum for discussing the latest research on database systems and advanced applications. DASFAA 2019 provided a forum for technical presentations and discussions among database researchers, developers, and users from academia, business, and industry, which showcases state-of-the-art R&D activities in database systems and their applications. The conference's long history has established the event as the premier research conference in the database area.

On behalf of the DASFAA 2019 program co-chairs, we are pleased to welcome you to the proceedings of the 24th International Conference on Database Systems for Advanced Applications DASFAA 2019, held during April 22–25, 2019, in Chiang Mai, Thailand. Chiang Mai is the largest city in northern Thailand. It is the capital of Chiang Mai Province and was a former capital of the kingdom of Lan Na 1296–1768, which later became the Kingdom of Chiang Mai, a tributary state of Siam from 1774 to 1899, and finally the seat of princely rulers until 1939. It is 700 km north of Bangkok near the highest mountains in the country. The city sits astride the Ping River, a major tributary of the Chao Phraya River.

We received 501 research paper submissions, each of which was assigned to at least three Program Committee (PC) members and one SPC member. The thoughtful discussion on each paper by the PC with facilitation and meta-review provided by the SPC resulted in the selection of 92 full research papers (acceptance ratio of 18%) and 64 short papers (acceptance ratio of 28%). In addition, we included 13 demo papers and six tutorials in the program. This year the dominant topics for the selected papers included big data, machine learning, graph and social network, recommendation, data integration and crowd sourcing, and spatial data management.

Three workshops are selected by the workshop co-chairs to be held in conjunction with DASFAA 2019, including BDMS: the 6th International Workshop on Big Data Management and Service; BDQM: the 4th Workshop on Big Data Quality Management; GDMA: the Third International Workshop on Graph Data Management and Analysis. We received 26 workshop paper submissions and accepted 14 papers.

The conference program included three keynote presentations by Prof. Anthony K. H. Tung National University of Singapore, Prof. Lei Chen The Hong Kong University of Science and Technology, and Prof. Ashraf Aboulnaga Qatar Computing Research Institute.

We wish to thank everyone who helped with the organization including the chairs of each workshop, demonstration chairs, and tutorial chairs and their respective PC members and reviewers. We thank all the authors who submitted their work, all of which contributed to making this part of the conference a success. We are grateful to the general chairs, Xue Li from The University of Queensland, Australia, and Nat Vorayos from Chiang Mai University, Thailand. We thank the local Organizing Committee chairs, Juggapong Natwichai and Krit Kwanngern from Chiang Mai

University, Thailand, for their tireless work before and during the conference. Special thanks go to the proceeding chairs, Yongxin Tong Beihang University, China and Juggapong Natwichai Chiang Mai University, Thailand, for producing the proceedings.

We hope that you will find the proceedings of DASFAA 2019 interesting and beneficial to your research.

March 2019

Guoliang Li  
Joao Gama  
Jun Yang

# Organization

## Honorary Co-chairs

Sampan Singharajwarapan	Chiang Mai University, Thailand
Bannakij Lojanapiwat	Chiang Mai University, Thailand

## General Co-chairs

Xue Li	The University of Queensland, Australia
Nat Vorayos	Chiang Mai University, Thailand

## Program Co-chairs

Guoliang Li	Tsinghua University, China
Joao Gama	University of Porto, Portugal
Jun Yang	Duke University, USA

## Industrial Track Co-chairs

Cong Yu	Google Inc., USA
Fan Jiang	Alibaba, China

## Workshop Co-chairs

Qun Chen	Northwestern Polytechnical University, China
Jun Miyazaki	Tokyo Institute of Technology, Japan

## Demo Track Co-chairs

Nan Tang	Qatar Computing Research Institute, Qatar
Masato Oguchi	Ochanomizu University, Japan

## Tutorial Chair

Sen Wang	The University of Queensland, Australia
----------	---

## Panel Chair

Lei Chen	Hong Kong University of Science and Technology, SAR China
----------	--

## **PhD School Co-chairs**

Sarana Nutanong	Vidyasirimedhi Institute of Science and Technology, Thailand
Xiaokui Xiao	National University of Singapore, Singapore

## **Proceedings Co-chairs**

Yongxin Tong	Beihang University, China
Juggapong Natwichai	Chiang Mai University, Thailand

## **Publicity Co-chairs**

Nah Yunmook	Dankook University, South Korea
Ju Fan	Renmin University of China, China
An Liu	Soochow University, China
Chiemi Watanabe	University of Tsukuba, Japan

## **Local Organization Co-chairs**

Juggapong Natwichai	Chiang Mai University, Thailand
Krit Kwangern	Chiang Mai University, Thailand

## **DASFAA Liaison**

Xiaofang Zhou	The University of Queensland, Australia
---------------	---

## **Regional Coordinator**

Pruet Boonma	Chiang Mai University, Thailand
--------------	---------------------------------

## **Web Master**

Titipat Sukhvibul	Chiang Mai University, Thailand
-------------------	---------------------------------

## **Senior Program Committee Members**

Philippe Bonnet	IT University of Copenhagen, Denmark
K. Selçuk Candan	Arizona State University, USA
Bin Cui	Peking University, China
Dong Deng	Rutgers University, USA
Yunjun Gao	Zhejiang University, China
Tingjian Ge	University of Massachusetts, Lowell, USA
Zhiguo Gong	Macau University, SAR China
Wook-Shin Han	Pohang University of Science and Technology, South Korea

Zi Huang	The University of Queensland, Australia
Wen-Chih Peng	National Chiao Tung University, Taiwan
Florin Rusu	UC Merced, USA
Chaokun Wang	Tsinghua University, China
Hongzhi Wang	Harbin Institute of Technology, China
Xiaokui Xiao	National University of Singapore, Singapore
Nan Zhang	The George Washington University, USA

## Program Committee

Alberto Abelló	Universitat Politècnica de Catalunya, Spain
Marco Aldinucci	University of Turin, Italy
Laurent Anne	University of Montpellier, LIRMM, CNRS, France
Akhil Arora	École polytechnique fédérale de Lausanne, Switzerland
Zhifeng Bao	Royal Melbourne Institute of Technology, Australia
Ladjel Bellatreche	ISAE-ENSMA, France
Yi Cai	South China University of Technology, China
Andrea Cali	Birkbeck University of London, UK
Yang Cao	Kyoto University, Japan
Yang Cao	University of Edinburgh, UK
Huiping Cao	New Mexico State University, USA
Lei Cao	Massachusetts Institute of Technology, USA
Barbara Catania	University of Genoa, Italy
Chengliang Chai	Tsinghua University, China
Lijun Chang	The University of Sydney, Australia
Cindy Chen	University of Massachusetts, Lowell, USA
Yueguo Chen	Renmin University of China, China
Reynold Cheng	The University of Hong Kong, SAR China
Fei Chiang	McMaster University, Canada
Linyang Chu	Simon Fraser University, Australia
Antonio Corral	University of Almeria, Spain
Lizhen Cui	Shandong University, China
Hasan Davulcu	Arizona State University, USA
Sabrina De Capitani di Vimercati	Università degli Studi di Milano, Italy
Zhiming Ding	Institute of Software, Chinese Academy of Sciences, China
Eduard Dragut	Temple University, USA
Lei Duan	Sichuan University, China
Amr Ebaid	Purdue University, USA
Ahmed Eldawy	University of California, Riverside, USA
Damiani Ernesto	University of Milan, Italy
Liyue Fan	University at Albany SUNY, USA
Ju Fan	Renmin University of China, China
Elena Ferrari	University of Insubria, Italy
Yanjie Fu	Missouri University of Science and Technology, USA



Xiaofeng Gao	Shanghai Jiaotong University, China
Yunjun Gao	Zhejiang University, China
Boris Glavic	Illinois Institute of Technology, USA
Neil Gong	Iowa State University, USA
Yu Gu	Northeastern University, China
Donghai Guan	Nanjing University of Aeronautics and Astronautics, China
Shuang Hao	Beijing Jiaotong University, China
Yeye He	Microsoft Research, USA
Huiqi Hu	East China Normal University, China
Juhua Hu	University of Washington, USA
Chao Huang	University of Notre Dame, USA
Matteo Interlandi	Microsoft, USA
Saiful Islam	Griffith University, Australia
Peiquan Jin	University of Science and Technology of China, China
Cheqing Jin	East China Normal University, China
Latifur Khan	The University of Texas at Dallas, USA
Peer Kroger	Ludwig-Maximilians-Universität München, Germany
Jae-Gil Lee	Korea Advanced Institute of Science and Technology, South Korea
Sang Won Lee	Sungkyunkwan University, South Korea
Young-Koo Lee	Kyung Hee University, South Korea
Zhixu Li	Soochow University, China
Lingli Li	Heilongjiang University, China
Jianxin Li	Deakin University, Australia
Jian Li	Tsinghua University, China
Cuiping Li	Renmin University of China, China
Qingzhong Li	Shandong University, China
Zheng Li	Amazon, USA
Bohan Li	Nanjing University of Aeronautics and Astronautics, China
Guoliang Li	Tsinghua University, China
Xiang Lian	Kent State University, USA
Chunbin Lin	Amazon AWS, USA
An Liu	Soochow University, China
Qing Liu	Data61, CSIRO, Australia
Eric Lo	Chinese University of Hong Kong, SAR China
Guodong Long	University of Technology Sydney, Australia
Cheng Long	Nanyang Technological University, Singapore
Hua Lu	Aalborg University, Denmark
Shuai Ma	Beihang University, China
Sofian Maabout	University of Bordeaux, France
Ioana Manolescu	Inria Saclay, France
Xiangfu Meng	Liaoning Technical University, China
Jun-Ki Min	Korea University of Technology and Education, South Korea

Yang-Sae Moon	Kangwon National University, South Korea
Parth Nagarkar	New Mexico State University, USA
Yunmook Nah	Dankook University, South Korea
Svetlozar Nestorov	Loyola University Chicago, USA
Quoc Viet Hung Nguyen	Griffith University, Australia
Baoning Niu	Taiyuan University of Technology, China
Kjetil Norvag	Norwegian University of Science and Technology, Norway
Sarana Yi Nutanong	City University of Hong Kong, SAR China
Vincent Oria	New Jersey Institute of Technology, USA
Xiao Pan	Shijiazhuang Tiedao University, China
Dhaval Patel	IBM TJ Watson Research Center, USA
Dieter Pfoser	George Mason University, USA
Silvestro Roberto Poccia	University of Turin, Italy
Weining Qian	East China Normal University, China
Tieyun Qian	Wuhan University, China
Shaojie Qiao	Chengdu University of Information Technology, China
Lu Qin	University of Technology Sydney, Australia
Xiaolin Qin	Nanjing University of Aeronautics and Astronautics, China
Weixiong Rao	Tongji University, China
Oscar Romero	Universitat Politècnica de Catalunya, Spain
Sourav S. Bhowmick	Nanyang Technological University, Singapore
Babak Salimi	University of Washington, USA
Simonas Saltenis	Aalborg University, Denmark
Claudio Schifanella	University of Turin, Italy
Marco Serafini	University of Massachusetts Amherst, USA
Zechao Shang	University of Chicago, USA
Xuequn Shang	Northwestern Polytechnical University, China
Shuo Shang	King Abdullah University of Science and Technology, Saudi Arabia
Wei Shen	Nankai University, China
Yanyan Shen	Shanghai Jiao Tong University, China
Kyuseok Shim	Seoul National University, South Korea
Alkis Simitsis	Hewlett Packard Enterprise, USA
Rohit Singh	Uber AI Labs, USA
Chunyao Song	Nankai University, China
Shaoxu Song	Tsinghua University, China
Weiwei Sun	Fudan University, China
Hailong Sun	Beihang University, China
Na Ta	Renmin University of China, China
Jing Tang	National University of Singapore, Singapore
Chaogang Tang	China University of Mining and Technology, China
Nan Tang	Qatar Computing Research Institute, Qatar
Shu Tao	IBM Research, USA

Saravanan Thirumuruganathan	Qatar Computing Research Institute, Qatar
Yongxin Tong	Beihang University, China
Hanghang Tong	Arizona State University, USA
Ismail Toroslu	Middle East Technical University, Turkey
Vincent Tseng	National Chiao Tung University, Taiwan
Leong Hou U.	University of Macau, SAR China
Qian Wang	Yanshan University, China
Sibo Wang	The Chinese University of Hong Kong, SAR China
Xiaoling Wang	East China Normal University, China
Xin Wang	Tianjin University, China
Ning Wang	Beijing Jiaotong University, China
Sen Wang	The University of Queensland, Australia
Jiannan Wang	Simon Fraser University, Australia
Jianmin Wang	Tsinghua University, China
Wei Wang	National University of Singapore, Singapore
Shi-ting Wen	Zhejiang University, China
Yinghui Wu	Washington State University, USA
Kesheng Wu	Lawrence Berkeley National Lab, USA
Mingjun Xiao	University of Science and Technology of China, China
Xike Xie	University of Science and Technology of China, China
Jianliang Xu	Hong Kong Baptist University, SAR China
Guandong Xu	University of Technology Sydney, Australia
Jianqiu Xu	Nanjing University of Aeronautics and Astronautics, China
Yajun Yang	Tianjin University, China
Yu Yang	Simon Fraser University, Australia
Bin Yao	Shanghai Jiaotong University, China
Lina Yao	University of New South Wales, Australia
Minghao Yin	Northeast Normal University, China
Hongzhi Yin	The University of Queensland, Australia
Man Lung Yiu	Hong Kong Polytechnic University, SAR China
Zhiwen Yu	South China University of Technology, China
Minghe Yu	Northeast University, China
Xiangyao Yu	Massachusetts Institute of Technology, USA
Jeffrey Xu Yu	Chinese University of Hong Kong, SAR China
Yi Yu	National Institute of Informatics, Japan
Ge Yu	Northeast University, China
Ye Yuan	Beijing Institute of Technology, China
Xiaowang Zhang	Tianjin University, China
Wei Zhang	East China Normal University, China
Yan Zhang	Peking University, China
Dongxiang Zhang	University of Electronic Science and Technology of China, China
Richong Zhang	Beihang University, China
Detian Zhang	Soochow University, China

Rui Zhang	University of Melbourne, Australia
Ying Zhang	University of Technology Sydney, Australia
Wenjie Zhang	University of New South Wales, Australia
Yong Zhang	Tsinghua University, China
Xiang Zhao	National University of Defence Technology, China
Lei Zhao	Soochow University, China
Jun Zhao	Nanyang Technological University, Singapore
Kai Zheng	University of Electronic Science and Technology of China, China
Yudian Zheng	Twitter, USA
Bolong Zheng	Aalborg University, Denmark
Rui Zhou	Swinburne University of Technology, Australia
Xiangmin Zhou	Royal Melbourne Institute of Technology, Australia
Yongluan Zhou	University of Copenhagen, Denmark
Yuanyuan Zhu	Wuhan University, China
Yan Zhuang	Tsinghua University, China

## Demo Program Committee

Lei Cao	Massachusetts Institute of Technology, USA
Yang Cao	University of Edinburgh, UK
Lijun Chang	The University of Sydney, Australia
Yueguo Chen	Renmin University of China, China
Fei Chiang	McMaster University, Canada
Bolin Ding	Alibaba, USA
Eduard Dragut	Temple University, USA
Amr Ebaid	Purdue University, USA
Ahmed Eldawy	University of California, Riverside, USA
Miki Enoki	IBM Research, Japan
Yunjun Gao	Zhejiang University, China
Tingjian Ge	University of Massachusetts, Lowell, USA
Yeye He	Microsoft Research, USA
Matteo Interlandi	Microsoft, USA
Zheng Li	Amazon, USA
Jianxin Li	Deakin University, Australia
Shuai Ma	Beihang University, China
Hidemoto Nakada	National Institute of Advanced Industrial Science and Technology, Japan
Abdulhakim Qahtan	Qatar Computing Research Institute, Qatar
Lu Qin	University of Technology Sydney, Australia
Jorge-Arnulfo Quiane-Ruiz	Qatar Computing Research Institute, Qatar
Elkindi Rezig	Massachusetts Institute of Technology, USA
Marco Serafini	University of Massachusetts Amherst, USA
Zechao Shang	University of Chicago, USA
Giovanni Simonini	Massachusetts Institute of Technology, USA
Rohit Singh	Uber AI Labs, USA

Yeong-Tae Song	Towson University, USA
Atsuko Takefusa	National Institute of Informatics, Japan
Nan Tang	Qatar Computing Research Institute, Qatar
Jiannan Wang	Simon Fraser University, Australia
Steven Euijong Whang	Korea Advanced Institute of Science and Technology, South Korea
Yinghui Wu	Washington State University, USA
Saneyasu Yamaguchi	Kogakuin University, Japan
Xiaochun Yang	Northeastern University, China
Peixiang Zhao	Florida State University, USA
Kai Zheng	University of Electronic Science and Technology of China, China
Xiangmin Zhou	Royal Melbourne Institute of Technology, Australia
Rui Zhou	Swinburne University of Technology, Australia

### Additional Reviewers

Aljebreen, Abdullah	He, Lihong	Pareek, Harsh
Alserafi, Ayman	He, Chenkun	Raimundo, Felix
Banerjee, Prithu	Jiang, Jinling	Ren, Weilong
Behrens, Hans	Jiang, Qize	Rezig, Elkindi
Bingham, Eli	Joglekar, Manas	Shan, Zhangqing
Chen, Jinpeng	Jovanovic, Petar	Son, Siwoon
Chen, Xilun	Lai, Ziliang	Sun, Lin
Cheng, Zelei	Li, Pengfei	Vachery, Jithin
Colla, Davide	Li, Mao-Lin	Varela, Edgar Ceh
Ebaid, Amr	Li, Huan	Wang, Yi-Chia
Feng, Chuanwen	Li, Su	Wang, Jin
Galhotra, Sainyam	Liang, Yuan	Wang, Ting
Garg, Nandani	Liu, Xincheng	Wu, Hao
Gaur, Garima	Liu, Chris	Zhang, Liming
Gil, Myeong-Seon	Liu, Sicong	Zhang, Pengfei
Gkountouna, Olga	Liu, Yi	Zhang, Yu
Gong, Qixu	Lu, Baotong	Zhang, Xinyu
Gurukar, Saket	Mazuran, Mirjana	Zhao, Xujian
Guzewicz, Pawel	Meduri, Vamsi	Zhao, Rong
Han, Yunheng	Mu, Lin	Zheng, Kaiping
Hao, Yifan	Munir, Faisal	Zou, Kai

## Contents – Part II

### Machine Learning

An Approach Based on Bayesian Networks for Query Selectivity Estimation . . . . .	3
<i>Max Halford, Philippe Saint-Pierre, and Franck Morvan</i>	
An Exploration of Cross-Modal Retrieval for Unseen Concepts . . . . .	20
<i>Fangming Zhong, Zhikui Chen, and Geyong Min</i>	
Continuous Patient-Centric Sequence Generation via Sequentially Coupled Adversarial Learning . . . . .	36
<i>Lu Wang, Wei Zhang, and Xiaofeng He</i>	
DMMAM: Deep Multi-source Multi-task Attention Model for Intensive Care Unit Diagnosis . . . . .	53
<i>Zhenkun Shi, Wanli Zuo, Weitong Chen, Lin Yue, Yuwei Hao, and Shining Liang</i>	
Learning $k$ -Occurrence Regular Expressions with Interleaving. . . . .	70
<i>Yeting Li, Xiaolan Zhang, Jialun Cao, Haiming Chen, and Chong Gao</i>	
Learning from User Social Relation for Document Sentiment Classification. . . . .	86
<i>Kangzhi Zhao, Yong Zhang, Yan Zhang, Chunxiao Xing, and Chao Li</i>	
Reinforcement Learning to Diversify Top-N Recommendation . . . . .	104
<i>Lixin Zou, Long Xia, Zhuoye Ding, Dawei Yin, Jiaying Song, and Weidong Liu</i>	
Retweeting Prediction Using Matrix Factorization with Binomial Distribution and Contextual Information. . . . .	121
<i>Bo Jiang, Zhigang Lu, Ning Li, Jianjun Wu, Feng Yi, and Dongxu Han</i>	
Sparse Gradient Compression for Distributed SGD . . . . .	139
<i>Haobo Sun, Yingxia Shao, Jiawei Jiang, Bin Cui, Kai Lei, Yu Xu, and Jiang Wang</i>	
STDR: A Deep Learning Method for Travel Time Estimation. . . . .	156
<i>Jie Xu, Yong Zhang, Li Chao, and Chunxiao Xing</i>	
Using Fractional Latent Topic to Enhance Recurrent Neural Network in Text Similarity Modeling . . . . .	173
<i>Yang Song, Wenxin Hu, and Liang He</i>	

Efficiently Mining Maximal Diverse Frequent Itemsets . . . . . 191  
*Dingming Wu, Dexin Luo, Christian S. Jensen,  
and Joshua Zhexue Huang*

**Privacy and Graph**

Efficient Local Search for Minimum Dominating Sets in Large Graphs . . . . . 211  
*Yi Fan, Yongxuan Lai, Chengqian Li, Nan Li, Zongjie Ma, Jun Zhou,  
Longin Jan Latecki, and Kaile Su*

Multi-level Graph Compression for Fast Reachability Detection . . . . . 229  
*Shikha Anirban, Junhu Wang, and Md. Saiful Islam*

Multiple Privacy Regimes Mechanism for Local Differential Privacy . . . . . 247  
*Yutong Ye, Min Zhang, Dengguo Feng, Hao Li, and Jialin Chi*

Privacy Preserving Elastic Stream Processing with Clouds Using  
Homomorphic Encryption . . . . . 264  
*Arosha Rodrigo, Miyuru Dayarathna, and Sanath Jayasena*

Select the Best for Me: Privacy-Preserving Polynomial Evaluation  
Algorithm over Road Network . . . . . 281  
*Wei Song, Chengliang Shi, Yuan Shen, and Zhiyong Peng*

**Recommendation**

AdaCML: Adaptive Collaborative Metric Learning for Recommendation . . . . . 301  
*Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Jiajie Xu, Junhua Fang,  
Lei Zhao, Victor S. Sheng, and Zhiming Cui*

Adaptive Attention-Aware Gated Recurrent Unit for Sequential  
Recommendation . . . . . 317  
*Anjing Luo, Pengpeng Zhao, Yanchi Liu, Jiajie Xu, Zhixu Li, Lei Zhao,  
Victor S. Sheng, and Zhiming Cui*

Attention and Convolution Enhanced Memory Network  
for Sequential Recommendation . . . . . 333  
*Jian Liu, Pengpeng Zhao, Yanchi Liu, Jiajie Xu, Junhua Fang, Lei Zhao,  
Victor S. Sheng, and Zhiming Cui*

Attention-Based Neural Tag Recommendation . . . . . 350  
*Jiahao Yuan, Yuanyuan Jin, Wenyan Liu, and Xiaoling Wang*

Density Matrix Based Preference Evolution Networks  
for E-Commerce Recommendation . . . . . 366  
*Panpan Wang, Zhao Li, Xuming Pan, Donghui Ding, Xia Chen,  
and Yuexian Hou*

Multi-source Multi-net Micro-video Recommendation with Hidden Item Category Discovery . . . . .	384
<i>Jingwei Ma, Jiahui Wen, Mingyang Zhong, Weitong Chen, Xiaofang Zhou, and Jadwiga Indulska</i>	
Incorporating Task-Oriented Representation in Text Classification . . . . .	401
<i>Xue Lei, Yi Cai, Jingyun Xu, Da Ren, Qing Li, and Ho-fung Leung</i>	
Music Playlist Recommendation with Long Short-Term Memory . . . . .	416
<i>Huiping Yang, Yan Zhao, Jinfu Xia, Bin Yao, Min Zhang, and Kai Zheng</i>	
Online Collaborative Filtering with Implicit Feedback . . . . .	433
<i>Jianwen Yin, Chenghao Liu, Jundong Li, BingTian Dai, Yun-chen Chen, Min Wu, and Jianling Sun</i>	
Subspace Ensemble-Based Neighbor User Searching for Neighborhood-Based Collaborative Filtering . . . . .	449
<i>Zepeng Li and Li Zhang</i>	
Towards both Local and Global Query Result Diversification . . . . .	464
<i>Ming Zhong, Huanyu Cheng, Ying Wang, Yuanyuan Zhu, Tiejun Qian, and Jianxin Li</i>	
<b>Social Network</b>	
Structured Spectral Clustering of PurTree Data . . . . .	485
<i>Xiaojun Chen, Chao Guo, Yixiang Fang, and Rui Mao</i>	
Dynamic Stochastic Block Model with Scale-Free Characteristic for Temporal Complex Networks . . . . .	502
<i>Xunxun Wu, Pengfei Jiao, Yaping Wang, Tianpeng Li, Wenjun Wang, and Bo Wang</i>	
In Good Company: Efficient Retrieval of the Top- <i>k</i> Most Relevant Event-Partner Pairs . . . . .	519
<i>Dingming Wu, Yi Zhu, and Christian S. Jensen</i>	
Local Experts Finding Across Multiple Social Networks . . . . .	536
<i>Yuliang Ma, Ye Yuan, Guoren Wang, Yishu Wang, Delong Ma, and Pengjie Cui</i>	
SBRNE: An Improved Unified Framework for Social and Behavior Recommendations with Network Embedding . . . . .	555
<i>Weizhong Zhao, Huifang Ma, Zhixin Li, Xiang Ao, and Ning Li</i>	
User Intention-Based Document Summarization on Heterogeneous Sentence Networks . . . . .	572
<i>Hsiu-Yi Wang, Jia-Wei Chang, and Jen-Wei Huang</i>	



**Spatial**

A Hierarchical Index Structure for Region-Aware Spatial Keyword Search with Edit Distance Constraint . . . . .	591
<i>Junye Yang, Yong Zhang, Huiqi Hu, and Chunxiao Xing</i>	
Collective POI Querying Based on Multiple Keywords and User Preference . . . . .	609
<i>Dongjin Yu, Yiyu Wu, Chengfei Liu, and Xiaoxiao Sun</i>	
DPSCAN: Structural Graph Clustering Based on Density Peaks . . . . .	626
<i>Changfa Wu, Yu Gu, and Ge Yu</i>	
Efficient Processing of Spatial Group Preference Queries . . . . .	642
<i>Zhou Zhang, Peiquan Jin, Yuan Tian, Shouhong Wan, and Lihua Yue</i>	
Reverse-Auction-Based Competitive Order Assignment for Mobile Taxi-Hailing Systems. . . . .	660
<i>Hui Zhao, Mingjun Xiao, Jie Wu, An Liu, and Baoyi An</i>	
Top-K Spatio-Topic Query on Social Media Data . . . . .	678
<i>Lianming Zhou, Xuanhao Chen, Yan Zhao, and Kai Zheng</i>	

**Spatio-Temporal**

A Frequency-Aware Spatio-Temporal Network for Traffic Flow Prediction . . . . .	697
<i>Shunfeng Peng, Yanyan Shen, Yanmin Zhu, and Yuting Chen</i>	
Efficient Algorithms for Solving Aggregate Keyword Routing Problems . . . . .	713
<i>Qize Jiang, Weiwei Sun, Baihua Zheng, and Kunjie Chen</i>	
Perceiving Topic Bubbles: Local Topic Detection in Spatio-Temporal Tweet Stream . . . . .	730
<i>Junsha Chen, Neng Gao, Cong Xue, Chenyang Tu, and Daren Zha</i>	
Real-Time Route Planning and Online Order Dispatch for Bus-Booking Platforms. . . . .	748
<i>Hao Zhou, Yucen Gao, Xiaofeng Gao, and Guihai Chen</i>	
STL: Online Detection of Taxi Trajectory Anomaly Based on Spatial-Temporal Laws . . . . .	764
<i>Bin Cheng, Shiyou Qian, Jian Cao, Guangtao Xue, Jiadi Yu, Yanmin Zhu, Minglu Li, and Tao Zhang</i>	
Correction to: Database Systems for Advanced Applications. . . . .	C1
<i>Guoliang Li, Jun Yang, Joao Gama, Juggapong Natwichai, and Yongxin Tong</i>	
<b>Author Index</b> . . . . .	781

# Contents – Part I

## Big Data

Accelerating Real-Time Tracking Applications over Big Data Stream with Constrained Space . . . . .	3
<i>Guangjun Wu, Xiaochun Yun, Shupeng Wang, Ge Fu, Chao Li, Yong Liu, Binbin Li, Yong Wang, and Zhihui Zhao</i>	
A Frequency Scaling Based Performance Indicator Framework for Big Data Systems . . . . .	19
<i>Chen Yang, Zhihui Du, Xiaofeng Meng, Yongjie Du, and Zhiqiang Duan</i>	
A Time-Series Sockpuppet Detection Method for Dynamic Social Relationships . . . . .	36
<i>Wei Zhou, Jingli Wang, Junyu Lin, Jiacheng Li, Jizhong Han, and Songlin Hu</i>	
Accelerating Hybrid Transactional/Analytical Processing Using Consistent Dual-Snapshot. . . . .	52
<i>Liang Li, Gang Wu, Guoren Wang, and Ye Yuan</i>	
HSDS: An Abstractive Model for Automatic Survey Generation . . . . .	70
<i>Xiao-Jian Jiang, Xian-Ling Mao, Bo-Si Feng, Xiaochi Wei, Bin-Bin Bian, and Heyan Huang</i>	
PU-Shapelets: Towards Pattern-Based Positive Unlabeled Classification of Time Series . . . . .	87
<i>Shen Liang, Yanchun Zhang, and Jiangang Ma</i>	

## Clustering and Classification

Discovering Relationship Patterns Among Associated Temporal Event Sequences . . . . .	107
<i>Chao Han, Lei Duan, Zhangxi Lin, Ruiqi Qin, Peng Zhang, and Jyrki Nummenmaa</i>	
Efficient Mining of Event Periodicity in Data Series . . . . .	124
<i>Hua Yuan, Yu Qian, and Mengna Bai</i>	
EPPADS: An Enhanced Phase-Based Performance-Aware Dynamic Scheduler for High Job Execution Performance in Large Scale Clusters. . . . .	140
<i>Prince Hamandawana, Ronnie Mativenga, Se Jin Kwon, and Tae-Sun Chung</i>	

<b>Incremental Discovery of Order Dependencies on Tuple Insertions . . . . .</b>	<b>157</b>
<i>Lin Zhu, Xu Sun, Zijing Tan, Kejia Yang, Weidong Yang, Xiangdong Zhou, and Yingjie Tian</i>	
<b>Multi-view Spectral Clustering via Multi-view Weighted Consensus and Matrix-Decomposition Based Discretization . . . . .</b>	<b>175</b>
<i>Man-Sheng Chen, Ling Huang, Chang-Dong Wang, and Dong Huang</i>	
<b>SIRCS: Slope-intercept-residual Compression by Correlation Sequencing for Multi-stream High Variation Data . . . . .</b>	<b>191</b>
<i>Zixin Ye, Wen Hua, Liwei Wang, and Xiaofang Zhou</i>	
<b>Crowdsourcing</b>	
<b>Fast Quorum-Based Log Replication and Replay for Fast Databases . . . . .</b>	<b>209</b>
<i>Donghui Wang, Peng Cai, Weining Qian, and Aoying Zhou</i>	
<b>PDCS: A Privacy-Preserving Distinct Counting Scheme for Mobile Sensing . . . . .</b>	<b>227</b>
<i>Xiaochen Yang, Ming Xu, Shaojing Fu, and Yuchuan Luo</i>	
<b>Reinforced Reliable Worker Selection for Spatial Crowdsensing Networks . . .</b>	<b>244</b>
<i>Yang Wang, Junwei Lu, Jingxiao Chen, Xiaofeng Gao, and Guihai Chen</i>	
<b>SeqST-ResNet: A Sequential Spatial Temporal ResNet for Task Prediction in Spatial Crowdsourcing . . . . .</b>	<b>260</b>
<i>Dongjun Zhai, An Liu, Shicheng Chen, Zhixu Li, and Xiangliang Zhang</i>	
<b>Towards Robust Arbitrarily Oriented Subspace Clustering . . . . .</b>	<b>276</b>
<i>Zhong Zhang, Chongming Gao, Chongzhi Liu, Qinli Yang, and Junming Shao</i>	
<b>Truthful Crowdsensed Data Trading Based on Reverse Auction and Blockchain. . . . .</b>	<b>292</b>
<i>Baoyi An, Mingjun Xiao, An Liu, Guoju Gao, and Hui Zhao</i>	
<b>Data Integration</b>	
<b>Selective Matrix Factorization for Multi-relational Data Fusion . . . . .</b>	<b>313</b>
<i>Yuehui Wang, Guoxian Yu, Carlotta Domeniconi, Jun Wang, Xiangliang Zhang, and Maozu Guo</i>	
<b>Selectivity Estimation on Set Containment Search . . . . .</b>	<b>330</b>
<i>Yang Yang, Wenjie Zhang, Ying Zhang, Xuemin Lin, and Liping Wang</i>	

Typicality-Based Across-Time Mapping of Entity Sets in Document Archives . . . . .	350
<i>Yijun Duan, Adam Jatowt, Sourav S. Bhowmick, and Masatoshi Yoshikawa</i>	
Unsupervised Entity Alignment Using Attribute Triples and Relation Triples . . . . .	367
<i>Fuzhen He, Zhixu Li, Yang Qiang, An Liu, Guanfeng Liu, Pengpeng Zhao, Lei Zhao, Min Zhang, and Zhigang Chen</i>	
Combining Meta-Graph and Attention for Recommendation over Heterogenous Information Network . . . . .	383
<i>Chenfei Zhao, Hengliang Wang, Yuan Li, and Kedian Mu</i>	
Efficient Search of the Most Cohesive Co-located Community in Attributed Networks . . . . .	398
<i>Jiehuan Luo, Xin Cao, Qiang Qu, and Yaqiong Liu</i>	
<b>Embedding</b>	
A Weighted Word Embedding Model for Text Classification . . . . .	419
<i>Haopeng Ren, ZeQuan Zeng, Yi Cai, Qing Du, Qing Li, and Haoran Xie</i>	
Bipartite Network Embedding via Effective Integration of Explicit and Implicit Relations . . . . .	435
<i>Yaping Wang, Pengfei Jiao, Wenjun Wang, Chunyu Lu, Hongtao Liu, and Bo Wang</i>	
Enhancing Network Embedding with Implicit Clustering . . . . .	452
<i>Qi Li, Jiang Zhong, Qing Li, Zehong Cao, and Chen Wang</i>	
MDAL: Multi-task Dual Attention LSTM Model for Semi-supervised Network Embedding . . . . .	468
<i>Longcan Wu, Daling Wang, Shi Feng, Yifei Zhang, and Ge Yu</i>	
Net2Text: An Edge Labelling Language Model for Personalized Review Generation . . . . .	484
<i>Shaofeng Xu, Yun Xiong, Xiangnan Kong, and Yangyong Zhu</i>	
Understanding Information Diffusion via Heterogeneous Information Network Embeddings . . . . .	501
<i>Yuan Su, Xi Zhang, Senzhang Wang, Binxing Fang, Tianle Zhang, and Philip S. Yu</i>	

**Graphs**

Distributed Parallel Structural Hole Detection on Big Graphs . . . . . 519  
*Faming Li, Zhaonian Zou, Jianzhong Li, Yingshu Li, and Yubiao Chen*

DynGraphGAN: Dynamic Graph Embedding via Generative Adversarial Networks . . . . . 536  
*Yun Xiong, Yao Zhang, Hanjie Fu, Wei Wang, Yangyong Zhu, and Philip S. Yu*

Evaluating Mixed Patterns on Large Data Graphs Using Bitmap Views . . . . . 553  
*Xiaoying Wu, Dimitri Theodoratos, Dimitrios Skoutas, and Michael Lan*

Heterogeneous Information Network Hashing for Fast Nearest Neighbor Search . . . . . 571  
*Zhen Peng, Minnan Luo, Jundong Li, Chen Chen, and Qinghua Zheng*

Learning Fine-Grained Patient Similarity with Dynamic Bayesian Network Embedded RNNs . . . . . 587  
*Yanda Wang, Weitong Chen, Bohan Li, and Robert Boots*

Towards Efficient  $k$ -TriPeak Decomposition on Large Graphs. . . . . 604  
*Xudong Wu, Long Yuan, Xuemin Lin, Shiyu Yang, and Wenjie Zhang*

**Knowledge Graph**

Evaluating the Choice of Tags in CQA Sites . . . . . 625  
*Rohan Banerjee, Sailaja Rajanala, and Manish Singh*

Fast Maximal Clique Enumeration for Real-World Graphs . . . . . 641  
*Yinuo Li, Zhiyuan Shao, Dongxiao Yu, Xiaofei Liao, and Hai Jin*

Leveraging Knowledge Graph Embeddings for Natural Language Question Answering . . . . . 659  
*Ruijie Wang, Meng Wang, Jun Liu, Weitong Chen, Michael Cochez, and Stefan Decker*

Measuring Semantic Relatedness with Knowledge Association Network. . . . . 676  
*Jiapeng Li, Wei Chen, Binbin Gu, Junhua Fang, Zhixu Li, and Lei Zhao*

SINE: Side Information Network Embedding . . . . . 692  
*Zitai Chen, Tongzhao Cai, Chuan Chen, Zibin Zheng, and Guohui Ling*

A Knowledge Graph Enhanced Topic Modeling Approach for Herb Recommendation . . . . . 709  
*Xinyu Wang, Ying Zhang, Xiaoling Wang, and Jin Chen*

Knowledge Base Error Detection with Relation Sensitive Embedding . . . . .	725
<i>San Kim, Xiuxing Li, Kaiyu Li, Jianhua Feng, Yan Huang, and Songfan Yang</i>	
Leon: A Distributed RDF Engine for Multi-query Processing . . . . .	742
<i>Xintong Guo, Hong Gao, and Zhaonian Zou</i>	
MathGraph: A Knowledge Graph for Automatically Solving Mathematical Exercises . . . . .	760
<i>Tianyu Zhao, Yan Huang, Songfan Yang, Yuyu Luo, Jianhua Feng, Yong Wang, Haitao Yuan, Kang Pan, Kaiyu Li, Haoda Li, and Fu Zhu</i>	
Multi-hop Path Queries over Knowledge Graphs with Neural Memory Networks . . . . .	777
<i>Qinyong Wang, Hongzhi Yin, Weiqing Wang, Zi Huang, Guibing Guo, and Quoc Viet Hung Nguyen</i>	
Sentiment Classification by Leveraging the Shared Knowledge from a Sequence of Domains . . . . .	795
<i>Guangyi Lv, Shuai Wang, Bing Liu, Enhong Chen, and Kun Zhang</i>	
<b>Author Index</b> . . . . .	813

# **Machine Learning**



# An Approach Based on Bayesian Networks for Query Selectivity Estimation

Max Halford<sup>1,2(✉)</sup>, Philippe Saint-Pierre<sup>2</sup>, and Franck Morvan<sup>1</sup>

<sup>1</sup> IRIT Laboratory, Paul Sabatier University, Toulouse, France  
{max.halford, frank.morvan}@irit.fr

<sup>2</sup> IMT Laboratory, Paul Sabatier University, Toulouse, France  
philippe.saint-pierre@math.univ-toulouse.fr

**Abstract.** The efficiency of a query execution plan depends on the accuracy of the selectivity estimates given to the query optimiser by the cost model. The cost model makes simplifying assumptions in order to produce said estimates in a timely manner. These assumptions lead to selectivity estimation errors that have dramatic effects on the quality of the resulting query execution plans. A convenient assumption that is ubiquitous among current cost models is to assume that attributes are independent with each other. However, it ignores potential correlations which can have a huge negative impact on the accuracy of the cost model. In this paper we attempt to relax the attribute value independence assumption without unreasonably deteriorating the accuracy of the cost model. We propose a novel approach based on a particular type of Bayesian networks called Chow-Liu trees to approximate the distribution of attribute values inside each relation of a database. Our results on the TPC-DS benchmark show that our method is an order of magnitude more precise than other approaches whilst remaining reasonably efficient in terms of time and space.

**Keywords:** Query optimisation · Cardinality estimation · Bayesian networks

## 1 Introduction

During query processing [34], each query goes through an optimisation phase followed by an execution phase. The objective of the optimisation phase is to produce an efficient query execution plan in a very short amount of time. The query optimiser draws on the cardinality estimates produced by the cost model for each relational operator in a given plan. Bad cardinality estimates propagate exponentially and have dramatic effects on query execution time [15]. Cardinality estimates are usually made based on a set of statistics collected from the relations and stored in the database's metadata. Such statistics are kept simple in order



to satisfy the limited time budget the query optimiser is allocated. However they usually don't capture attribute dependencies.

Formally, given a query  $\mathcal{Q}(\mathcal{R}, \mathcal{J}, \mathcal{A})$  over a set of relations  $\mathcal{R}$ , a set of join predicates  $\mathcal{J}$  and a set of attribute predicates  $\mathcal{A}$ , the cardinality of the query is computed as follows:

$$|\mathcal{Q}(\mathcal{R}, \mathcal{J}, \mathcal{A})| = P(\mathcal{J}, \mathcal{A}) \times \prod_{R \in \mathcal{R}} |R| \quad (1)$$

where  $P(\mathcal{J}, \mathcal{A})$  is the selectivity of the query and  $\prod_{R \in \mathcal{R}} |R|$  is the number of tuples in the Cartesian product of the involved relations. The problem is that  $P(\mathcal{J}, \mathcal{A})$  is not available. Moreover estimating it quickly leads to a combinatorial explosion. Simplifying assumptions are made in order to approximate the selectivity whilst ensuring a realistic computational complexity [34].

The first assumption that is commonly made is that attributes are independent within and between each relation. This is the so-called *attribute value independence* (AVI) assumption. It allows to simplify the computation as follows:

$$P(\mathcal{A}) \simeq \prod_{A_R \in \mathcal{A}} P(A_R) \simeq \prod_{A_R \in \mathcal{A}} \prod_{a_i \in A_R} P(a_i) \quad (2)$$

where  $P(A_R)$  refers to the selectivity concerning relation  $R$  whilst  $P(a_i)$  stands for the selectivity of a predicate on attribute  $a_i$ . In practice the AVI assumption is very error-prone because attributes often exhibit correlations. However it is extremely practical because each distribution  $P(a_i)$  can be condensed into a one-dimensional histogram  $\tilde{P}(a_i)$ .

Next, the *join predicate independence* assumption implies that join selectivities can be computed independently, which leads to the following approximation:

$$P(\mathcal{J}) \simeq \prod_{J_i \in \mathcal{J}} P(J_i) \quad (3)$$

Assume we are given two relations  $R$  and  $S$ . We want to join both relations on their respective attributes  $R.K$  and  $S.F$ . In this case the selectivity of the join (denoted  $J$ ) can be computed exactly [34]:

$$P(J) = \min\left(\frac{1}{|J.R.K|}, \frac{1}{|J.S.F|}\right) \quad (4)$$

The previous assumption doesn't usually hold if multiple foreign keys are included in a join [15]. Finally, the *join uniformity* assumption states that attributes preserve their distributions after joins. This allows the following simplification:

$$P(\mathcal{J}, \mathcal{A}) \simeq P(\mathcal{J}) \times P(\mathcal{A}) \quad (5)$$

Most relational databases [2, 13, 36] assume all the previous assumptions in conjunction, which leads to the following formula:

$$P(\mathcal{J}, \mathcal{A}) \simeq \prod_{J_i \in \mathcal{J}} \min\left(\frac{1}{|J_i.R.K|}, \frac{1}{|J_i.S.F|}\right) \times \prod_{A_R \in \mathcal{A}} \prod_{a_i \in A_R} \tilde{P}(a_i) \quad (6)$$

In practice the previous approximation is much too coarse and is frequently wrong by orders of magnitude. However it only requires storage space that grows linearly with the number of attributes and doesn't involve any prohibitive computation. In other words accurate cardinality estimation is traded in exchange for a low computational complexity. The natural question is if a better trade-off is possible. That is, one that relaxes any of the previous assumptions.

A lot of work has gone into developing *attribute-level* synopses [12, 32] which approximate the distribution  $P(a)$  of each attribute  $a$ . Mostly this involves using histograms and other well-studied statistical constructs. Although theoretically sound, these methods do not help in handling commonplace queries that involve more than one attribute predicate. Furthermore, *table-level* synopses [27] have been proposed to capture dependencies between attributes. The problem is that methods of this kind, such as multi-dimensional histograms, usually require an amount of storage space that grows exponentially with the number of attributes. Table-level synopses also includes various sampling methods [25, 30, 38] where the idea is simply to execute a query on a sample of the database and extrapolate the cardinality. Although they don't handle dependencies across relations, they are computationally efficient because they don't require joins. Finally, *schema-level* synopses [6, 19, 23, 38] attempt to soften the join uniformity and join predicate independence assumptions. Although these methods have the potential to handle join-crossing correlations [21], they require a prohibitive amount of computational resources because of the amount of joins they necessitate.

Accurate schema-level methods based on Bayesian networks have been proposed [10, 37]. A Bayesian network factorises a distribution in order to represent it with a product of lower dimensional distributions. Each lower dimensional distribution captures a dependency between two or more attributes. For example the distribution  $P(\text{hair}, \text{nationality})$  can be factorised as  $P(\text{hair}|\text{nationality}) \times P(\text{nationality})$  because a person's hair colour is correlated with his nationality. The trick is that finding the right factorisation is an NP-hard problem [18]. Moreover the time required to produce estimates increases with the complexity of the factorisation [33]. The method proposed in [10] successfully captures attribute dependencies across relations but it requires a prohibitive amount of computational complexity that makes it unusable in practice. [37] propose a simpler method that only attempts to capture dependencies between two relations at most. Although their proposal is more efficient, it still requires performing a significant amount of joins. Moreover the factorisation structures used in both proposals incur an inference procedure that doesn't run in linear time. We believe that giving up some of the accuracy of existing proposals leads to methods that strike a better balance between accuracy and computational complexity. To this extent we propose to factorise the distribution of attributes only inside each relation. We argue that having reliable selectivity estimates for single relations is fundamental for estimating the size of joins [15]. Furthermore we propose to extend a particular type of Bayesian networks called Chow-Liu trees. These allow us to use network structures that are efficient space-wise and can be queried in sub-linear time. Although our approach doesn't capture as many dependencies

as in [10] and [37], it can be compiled quicker and can produce selectivity estimates in less time. Moreover it is still an order of magnitude more precise than trivial models that assume independence.

The rest of this paper is organised as follows. Section 2 gives an overview of existing methods and their associated pros and cons. This is also where we introduce some notions relating to Bayesian networks. Section 3 is where we describe our model and show how it can efficiently be used for the task of selectivity estimation. Section 4 compares our model to PostgreSQL’s cost engine and to a Bernoulli sampling estimator on the TPC-DS benchmark. We explain in what cases our model succeeds and in what cases it doesn’t bring anything to the table. Finally, Sect. 5 concludes and points to some research opportunities.

## 2 Related Work

### 2.1 Distribution Estimation

The most prominent approach in cost-based query optimisation is to approximate the distribution of attributes of a given database. This has been an area of research ever since *equi-width histograms* were used for summarising a single attribute [20]. *Equi-height histograms* are commonly used because of their provably lower average error [30]. Meanwhile [16] showed that histograms that minimise the average selectivity estimation error are ones that minimise variance inside each bucket. These histograms are usually called *V-optimal histograms* and involve a prohibitive mathematical optimisation process. As a compromise, [16] introduced the notion of *biased histograms* to find a balance between memorising exact frequencies and approximating them. Histograms are well understood in theory and ubiquitously used in practice, however they don’t capture dependencies between attributes.

Multi-column histograms [3, 12, 27] have been proposed to handle dependencies between two or more attributes. Although they are sound in theory, in practice they are difficult to build and even more so to update [28]. Moreover they require storage space that grows exponentially with the number of attributes.

To mitigate the exponential growth problem of multi-dimensional histograms, one approach is use a factorised representation of a distribution. The idea is to represent a distribution  $P(A_1, \dots, A_n)$  as a product of smaller conditional distributions  $P(A_i | \text{Parents}(A_i))$ . For example the distribution  $P(A_1, A_2, A_3)$  can be estimated as  $\hat{P}(A_1, A_2, A_3) = P(A_1 | A_2)P(A_3 | A_2)P(A_3)$ .  $\hat{P}(A_1, A_2, A_3)$  is necessarily an approximation because it doesn’t capture the three-way interaction between  $A_1$ ,  $A_2$ , and  $A_3$ . The benefit is that although  $\hat{P}(A_1, A_2, A_3)$  is an approximation, it requires less storage space. Moreover if  $A_1$  and  $A_3$  are independent then no information is lost. Bayesian networks [18] have been shown to be a strong method to find such approximations. However, querying a BN is an NP-hard problem [8] and can take a prohibitive amount of time depending on the structure of the network. Moreover, off-the-shelf implementations don’t restrict the structure of the final approximation. This leads to approximations which either require a prohibitive amount of storage space, or are too slow, or both.

BNs are classically studied in the context of a single tabular dataset. However in a relational database the data is contained in multiple relations that share relationships. [10] first introduced *probabilistic relational models* that could handle the relational setting. They introduced the notion of a *join indicator* to relax the join uniformity assumption. However for structure learning and inference they use off-the-shelf algorithms with running complexities that are way too prohibitive for a database context. [37] extended this work and proposed to restrict the dependencies a BN can capture to be between two relations at most. Even though their procedure is more efficient, it still requires joining relations, albeit only two at once. Both of these proposals work at a schema-level and require performing a prohibitive amount of joins. Although existing methods based on Bayesian networks seem promising, we argue that they are still too complex to be used at a large scale.

The problem of learning distributions is that they inescapably require a lot of storage space. A radically different approach that has made it's mark is to execute the query on a sample of the database in order to extrapolate the query's cardinality.

## 2.2 Sampling

Sampling is most commonly used to estimate selectivity for a query that pertains to a single relation [30]. The simplest method is to sample a relation  $R_i$  with probability  $p_i$ . The obtained sample  $r_i$  will then contain  $|R_i| \times p_i$  tuples. To estimate the selectivity of a query on  $R_i$  one may run the query on it's associated sample  $r_i$  and multiply the cardinality of the output by  $\frac{1}{p}$ . This is commonly referred to as *Bernoulli sampling* and works rather well given a sufficient sample size. Adaptive methods [25] have also been proposed to determine an optimal sample size for each relation. Sampling is attractive because it is simple to implement and naturally captures dependencies between attributes. Moreover, sampling can be performed on multiple relations in order to capture inter-relational attribute dependencies.

Sampling across multiple relations is a difficult task. Indeed [4] showed that the join of independent uniform samples of relations is not a uniform sample of the join of the relations. Many methods have been proposed for two-way joins [29, 38]. Their common idea is to use a hash function  $h(a) \rightarrow [0, 1]$  to make sure joined samples share keys. Say  $R_1$  has an attribute  $A_1$  which is a foreign key to an attribute  $A_2$  of a relation  $R_2$ . By applying the same hash function  $h(a)$  to both attributes, one may obtain samples which preserve join relationships by keeping all the tuples that satisfy  $h(a) < p$ . This way all tuples from both relations that satisfy  $h(a) < p$  will be included in the sample. The unbiased estimator for the size of the result of the join is  $J(R_1, R_2)$  is  $\frac{J(r_1, r_2)}{p}$ . Although quite strong in theory, join-aware sampling [24] requires a prohibitive full pass over the involved relations if no index is available. Moreover, this approach doesn't necessarily extend to joins involving more than two relations [1].

## 2.3 Learning

To completely sidestep the difficulties inherent to query optimisation, learning procedures that correct their mistakes have been proposed [5, 19, 35]. In the case of database optimisation learning has been used to tune various models and to memorise observed selectivities. This is done by *query feedback* where the cost model gets access to the actual cardinalities [5] after the query execution phase. By comparing the estimates it has made with the actual values it can make adjustments with the goal of making less mistakes for subsequent queries. The most successful method in this category is DB2’s LEO optimiser [35]. The approach LEO takes is simply to memorise the true cardinality of error-prone parts of executed query plans. This works remarkably well in an environment where a given query is run repeatedly. However it doesn’t help for estimating the cardinality of unseen queries. Recently an interesting approach based on deep learning has also been proposed [19]. Apart from LEO, learning approaches have not yet matured enough to be used in practice.

## 2.4 Discussion

All of the previously mentioned methods offer a different compromise in terms of accurate cardinality estimation, temporal complexity, and spatial complexity. On the one hand, histograms are precise, quick, and lightweight. However, they require a prohibitive amount of storage space if one wishes to capture attribute dependencies. On the other hand, sampling can easily capture attribute dependencies; but it is too slow because either the sample has to be constructed online or loaded in memory. Finally learning is an original take on the problem but it doesn’t help for unseen queries. Our contribution is to use Bayesian networks to factorise the distribution of the attributes of each relation. This way we capture the most important attribute dependencies and ignore the unimportant ones to preserve storage space. A huge benefit of our method is that we can optimise each Bayesian network on a sample of the associated relation to save time without a significant loss in accuracy. The downside is that like most methods we ignore dependencies between attributes of different relations.

# 3 Methodology

## 3.1 Finding a Good Network

A Bayesian network (BN) factorises a probability distribution  $P(X_1, \dots, X_n)$  into a product of conditional distributions. For any given probability distribution  $P(\mathcal{X})$  there exist many possible BNs. For example  $P(\text{hair}, \text{nationality}, \text{gender})$  can be factorised as  $P(\text{hair}|\text{nationality})P(\text{gender}|\text{nationality})P(\text{nationality})$  as well as  $P(\text{hair}|(\text{nationality}, \text{gender}))P(\text{nationality})P(\text{gender})$  (see Fig. 1).



**Fig. 1.** Possible factorisations of  $P(\text{hair}, \text{nationality}, \text{gender})$

The goal of *structure learning* is to find a BN that closely matches  $P(\mathcal{X})$  whilst preserving a low computational complexity. Indeed, for any given BN, the cost of storing it and of computing a marginal distribution  $P(X_i)$  depend on its structure.

The classic approach to structure learning is to define a scoring function that determines how good a BN is – both in terms of accuracy and complexity – and to run a mathematical optimisation procedure over the possible structures [11]. The problem is that such kind of procedures are too costly and don't fit inside the tight computational budget a database typically imposes. Recently linear programming approaches that require an upper bound on the number of parents have also been proposed [17]; in practice these can handle problems with up to 100 variables which is far from ideal. Finally, one can also resort to using greedy algorithms that run in polynomial time but don't necessarily find a global optimum.

*Chow-Liu trees* [7] is one such method that finds a BN where dependencies between two attributes are the only ones considered. Building a Chow-Liu tree only involves three steps. Initially the mutual information (MI) between each pair of attributes is computed. These values define a fully connected graph  $G$  where each MI value is translated to a weighed edge. Next, a minimum spanning tree (MST) of  $G$  is retrieved. This can be done in  $\mathcal{O}(n \log(n))$  time where  $n$  is the number of attributes. Finally the MST has to be directed by choosing a node at random and defining it as the root.

We choose to use Chow-Liu trees for two practical reasons. First of all they are simple to construct. The only part that doesn't scale well is computing the MI values. However this can be accelerated by using a coarser representation of the data such as histograms. Moreover the process can be run over a sample of a relation. In our experience these two tricks greatly reduced computation time without hindering the accuracy of the resulting trees. Secondly the output network is a tree – hence there is only one parent per node. This is practical because retrieving a marginal distribution – in other words *inferring* – from a tree can be done in linear time [33]. Moreover, storing the network only requires saving  $n - 1$  two-dimensional distributions and one uni-dimensional distribution. On top of this, [7] proves that Chow-Liu trees minimise the KL divergence, meaning that they are the best possible trees from an information theory perspective. The downside is that they can't capture dependencies between more than 2 variables – for example it only snows if it's cold and rainy. However in our experience these kind of dependencies are not so common.

### 3.2 Estimating the Conditional Probabilities

Once a satisfying structure has been found, the necessary probability distributions have to be computed. Indeed recall that a Bayesian network is nothing more than a product of conditional probability distributions (CPD). A CPD gives the distribution of a variable given the value of one or more so-called parent variables (Table 1). For example Tables 2 and 3 are two CPDs that are both conditioned on the *nationality* variable.

**Table 1.**  $P(\textit{nationality})$

American	Swedish
0.5	0.5

**Table 2.**  $P(\textit{hair}|\textit{nationality})$

	Blond	Brown	Dark
American	0.2	0.6	0.2
Swedish	0.8	0.2	0

**Table 3.**  $P(\textit{gender}|\textit{nationality})$

	Male	Female
American	0.5	0.5
Swedish	0.45	0.55

The number of values needed to define a CPD is  $c^{p+1}$  where  $c$  is the cardinality of each variable – for simplicity we assume it is constant – and  $p$  is the number of parent variables. This stems from the fact that each CPD is related to  $p + 1$  variables and that each and every combination of values has to be accounted for. The fact that Chow-Liu trees limits the number of parents each node has to 1 means that we only have to store  $c^2$  values per distribution. Moreover a sparse representation can be used to leverage the fact that 0s are frequent. However, if the cardinality of a variable is high then a lot of values still have to be stored. This can be rather problematic in a constrained environment.

To preserve a low spatial complexity we propose to use end-biased histograms described in Subsect. 2.1. The idea is to preserve the exact probabilities for the  $k$  most common values of a variable and put the rest of the probabilities inside  $j$  equi-height intervals. Using equi-height intervals means that we don't have to store the frequency of each interval. Indeed it is simply  $1 - \sum_{i=1}^k P(MCV_i)$  where  $P(MCV_i)$  denotes the frequency of the  $i^{th}$  most common value. Instead, by assuming that the values inside an interval are uniformly distributed, we only have to store the number of distinct values the interval contains. Table 4 shows what a CPD with intervals looks like. In the example, given that a person is American, there is probability of  $1 - (0.2 + 0.5) = 0.3$  that his hair colour is in the [Dark, Red] interval. Because there are 3 distinct values in the [Dark, Red] interval, the probability that an American has, say, hazel hair is  $\frac{1-(0.2+0.5)}{3} = 0.1$ .

**Table 4.**  $P(\textit{hair}|\textit{nationality})$  with  $k = 2$  and  $j = 1$

	Blond	Brown	[Dark, Red]
American	0.2	0.5	3
[British, French]	0.4	0.3	3
Swedish	0.8	0.2	0

Compressing a CPD this way means we only have to store  $(k + j)^2$  values per distribution. If we assume that there are  $n$  attributes inside a relation then storing a Bayesian networks requires  $(k + j) + (n - 1)(k + j)^2$  values in total – the first  $(k + j)$  corresponds to the network’s root node which is not conditioned on any other variable. This has the added advantage that we can handle continuous variables that usually have extremely high cardinalities.

Fortunately, retrieving CPDs inside a relational database can easily be done with the `GROUP BY` and `COUNT` statements. Moreover, the CPDs can be computed on a sample of the relations to reduce computation time. Whats more, if data is appended to the database then only the CPDs have to recomputed if one assumes the structures of the Bayesian networks remain constant through time. However, if new attributes are added to a relation then the structure of it’s Bayesian network has to be rebuilt from scratch.

### 3.3 Producing Selectivity Estimates

As previously mentioned, inference is the task of obtaining a marginal distribution from a Bayesian network. For example we may want to know the probability of Swedish people having blond hair (i.e.  $P(\text{hair} = \text{“Blond”} \wedge \text{nationality} = \text{“Swedish”})$ ). The idea is to treat the obtained probability as the selectivity of the associated relational query. For each relation involved in a query, we identify the part of the query that applies to the relation and determine it’s selectivity. Then, by assuming that attributes from different relations are independent, we simply multiply the selectivities together. Although this is a strong assumption, we argue that capturing table-level dependencies can still have a significant impact on the overall cardinality estimation. Of course we would be even more precise if we had determined dependencies between different relations as in [10,37], but it would necessarily involve joins. In other words our method offers a different trade-off between accuracy and computational feasibility.

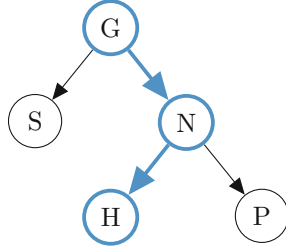
Performing inference over a BN is an NP-hard problem [8]. However, because we have restricted our BNs to trees, we can make full use of purpose-built algorithms that only apply to trees. The *variable elimination* (VE) algorithm [9] is a simple exact inference algorithm that can be applied to any kind of network topology. Specifically the complexity of VE is  $\mathcal{O}(n \exp(w))$  where  $n$  is the number of nodes and  $w$  is the width of the network [33]. However the width of a tree is necessarily 1, meaning VE can run in  $\mathcal{O}(n)$  time. The formula for applying VE is given in (7), wherein  $k$  attributes are being queried out of a total of  $n$ .

$$P(A_1 = a_1, \dots, A_k = a_k) = \sum_{i=k+1}^n \prod_{j=1}^k P(A_j = a_j | \text{Parents}(A_j)) \quad (7)$$

The idea of VE is to walk over the tree in a post-order fashion – i.e. start from the leaves – and sum up each CPD row-wise. This avoids unnecessarily computing sums more than needed and ensures the inference process runs in linear time. The computation can be further increased by noticing that not all



nodes in a BN are needed to obtain a given marginal distribution [18]. Indeed the VE algorithm only has to be run on a necessary subset of the tree’s nodes which is commonly referred to as the *Steiner tree* [14]. Extracting a Steiner tree from a tree can be done in linear time (see Algorithm 1).



**Fig. 2.** Steiner tree in blue containing nodes G, N, and H needed to compute H’s marginal distribution (Color figure online)

In our case we are using CPDs with intervals, meaning that we have to tailor the VE algorithm around them. Fortunately this is quite simple as we only have to check if a given value is an interval or not. Range queries can be handled by interpolating inside the interval whilst for equality queries we can assume that all distinct values in the interval are equally frequent (Fig. 2).

---

### Algorithm 1. Steiner tree extraction

---

```

1: function WALK(node, required, path, relevant)
2:   if required is empty then
3:     return {}
4:   else if node in nodes then
5:     required  $\leftarrow$  required  $\setminus$  {node}
6:     relevant  $\leftarrow$  relevant  $\cup$  path
7:   end if
8:   path  $\leftarrow$  path  $\cup$  {node}
9:   for child  $\in$  node.children() do
10:    relevant  $\leftarrow$  relevant  $\cup$  WALK(child, required, path, relevant)
11:  end for
12:  return relevant
13: end function

14: function EXTRACTSTEINERTREE(tree, nodes)
15:  nodes  $\leftarrow$  nodes  $\cup$  tree.root()
16:  relevant  $\leftarrow$  WALK(tree, nodes, {}, {})
17:  return tree.subset(relevant)
18: end function

```

---

## 4 Experimental Study

### 4.1 Setup

We implemented a prototype of our method along with the Bernoulli sampling described in Sect. 2.2 and the textbook method described in the introduction. We chose these two methods because they are realistic and are used in practice. We would have liked to compare our method to previous Bayesian approaches proposed in [10,37], however we were not able to accurately reproduce their results given the available information. We expect our method to be less accurate but much more computationally efficient. Our goal is to quantitatively show why our method offers a better trade-off than the other two implemented methods. We ran all methods against a small subset of the queries contained in the TPC-DS benchmark [31] with a scale factor of 20<sup>1</sup>. We only picked queries that apply more than one attribute predicate on at least one relation and that exhibit dependencies. We chose this subset on purpose because our model doesn't bring anything new to the table if there is only one predicate. Indeed if there is only one predicate then our model is equivalent to the textbook approach of using one histogram per attribute.

We used four criteria to compare each method: (1) The construction time. (2) The accuracy of the cardinality estimates. (3) The time needed to make a cardinality estimate. (4) The number of values needed to store the model. We ran our experiments multiple times to get statistically meaningful results. First of all we used 10 different sample sizes to determine the impact of sampling. Then, for each combination of method and sampling size we took measurements with 10 different seeds. For each measurement we thus calculated its mean and its standard deviation. To make the comparison fair we used equi-height histograms with the same parameters for both the textbook and the Bayesian networks approaches. Specifically we stored the exact frequencies of the 30 most common values and approximated the rest with 30 buckets.

### 4.2 Construction Time

We first of all measured the time it takes to construct each model (see Fig. 3). Naturally **sampling** is the method that takes the least time because nothing has to be done once the sample is retrieved from the database. The **textbook** and **Bayesian network** ours, that is) methods necessarily take longer because they have to perform additional computations after having obtained the sample. The **textbook** method only has to build equi-height histograms. The **Bayesian network** method requires slightly more involved calculations. It spends most of its time computing mutual information scores and processing **GROUP BY** operations. Although these operations are unavoidable, their running time can be mitigated as explained in Sect. 3.1. Moreover the **GROUP BY** results used for computing mutual information scores can be reused for parameter estimation as

---

<sup>1</sup> Specifically we used the following queries: 7, 13, 18, 26, 27, 53, 54, 91.

explained in Sect. 3.2. However for the sake of simplicity we didn’t implement these optimisations in our prototype. Still, the results we obtained seem better than those presented in [37] where the authors claim their method can process a database of 1 in about an hour. Our method can process the same amount of data in under 8 min.

### 4.3 Cardinality Estimates

We then compared methods based on their average accuracy. In other words we ran each method against each query and measured the average error. Although our method improves the accuracy of cardinality estimation for queries on a single relation; our goal in this benchmark is to measure how much this will impact the overall accuracy for general queries over multiple relations. It is possible that some queries bias the average accuracy because each query returns a number of rows that can vary in magnitude in regard to the others. Because of this, typical metrics such as the mean squared error (MSE) can’t be used. [26] explain why using average multiplicative errors makes the most sense in the context of query optimisation. For a given number of rows  $y$  and an estimate  $\hat{y}$  we calculated the  $q$ -error [22] which is defined as  $q(y, \hat{y}) = \frac{\max(y, \hat{y})}{\min(y, \hat{y})}$ .

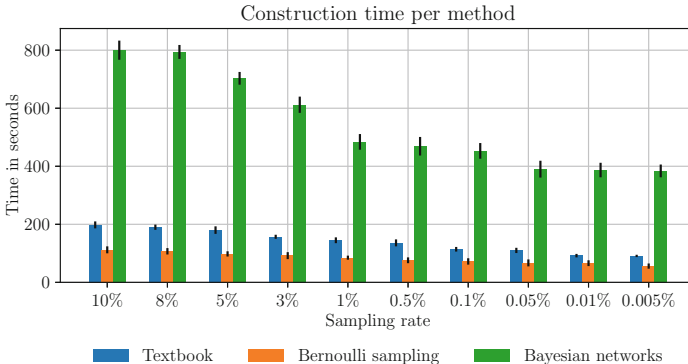
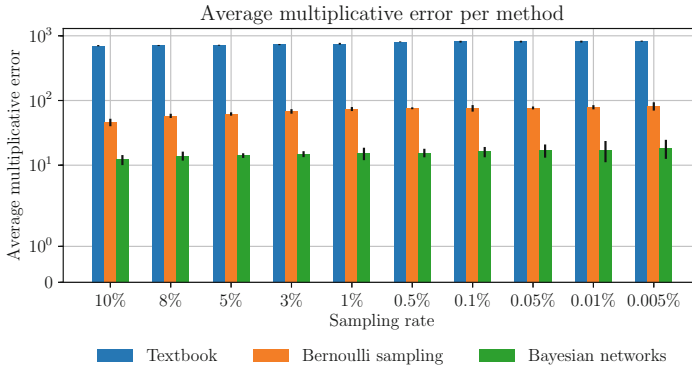


Fig. 3. Construction time

The advantage of the  $q$ -error is that it returns an error that is independent of the scale of the values at hand. Moreover the  $q$ -error is symmetric will thus be the same regardless of the fact that we are underestimating or overestimating the cardinality. For each combination of method and sampling rate we averaged the  $q$ -error over all 8 queries. As previously mentioned we took measurements with different samples so to reduce any possible bias in the results. The results are displayed in Fig. 4.

Unsurprisingly, the **textbook** method produces estimates that are off by several orders of magnitude. What is more interesting is that **Bayesian networks**

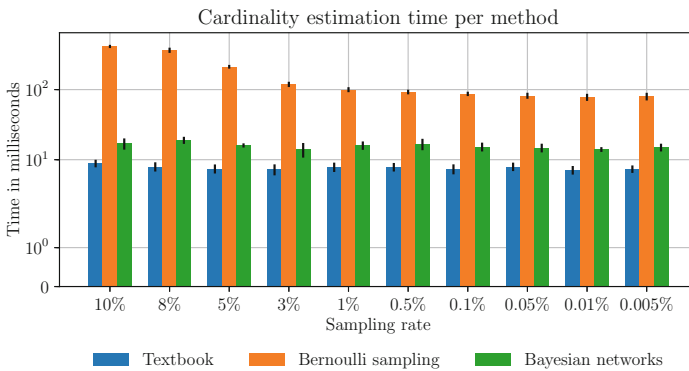


**Fig. 4.** Average errors

are significantly better than **sampling**. The reason this occurs is because the sampling method doesn't place any uncertainty as to if a value is present in a relation or not. A value is either in a sample or not. Meanwhile the Bayesian networks method uses histograms to approximate the frequencies of the least common values. This has a significant impact on the overall average, at least for the subset of queries we chose.

#### 4.4 Inference Time

We also measured the average time it takes to estimate the selectivity of a query. A query optimiser has a very small amount of time to produce a query execution plan. Only a fraction of this time can be allocated to cardinality estimation. It is thus extremely important to produce somewhat accurate cardinality estimates in a timely fashion. We recorded our results in Fig. 5.



**Fig. 5.** Cardinality estimation time

As can be seen the main pitfall of the **sampling** method is that it takes a considerable amount of time to estimate a cardinality. This is expected because for each set of predicates a full pass has to be made on the according sample. Whats more we didn't even take into account the fact that the necessary samples have to be loaded in memory beforehand. As for the **textbook** method, it only has to read values from a histogram. Meanwhile the **Bayesian networks** method has to extract the Steiner tree and perform variable elimination on it as explained in Sect. 3.3. This is naturally more burdensome than simply looking up values in a histogram, but it is still on the same order of magnitude in terms of time.

#### 4.5 Disk Usage

Finally we measured the number of values needed to store each model. For the sampling method each and every sample has to be stored. Meanwhile the textbook and Bayesian networks methods are synopses and require storing a significantly lesser amount of values. In our experiments the worse case storage bounds of both of these methods are pessimistic. For example in our experiments, the theoretical upper bound textbook method is around 32000 values, but only 53% of the values actually need to be stored (the other 47% are 0s). Moreover the same occurs for the Bayesian networks method; indeed for a 5% sample only around 300000 values out of the theoretical 400000 have to be stored. This is due to the fact that some attributes have a very low number of unique values which makes the associated histograms smaller than expected.

**Table 5.** Storage size per method using a 5% sampling rate

	Size	Sparsity	Effective size
Textbook	117 KB	47%	62 KB
Sampling	412 MB	0%	412 MB
Bayesian network	615 KB	24%	467.4 KB

The numbers presented in Table 5 were obtained by using a 5% sample of the database. Apart from the sampling method the numbers are more or less the same when using different sample sizes. Indeed histograms and conditional probability distributions have a fixed size which doesn't increase with the amount of data they synthesise. Meanwhile using sampling means that the whole has to be stored either in memory or on the disk. We noticed that the higher the dependencies between the attributes, the higher the sparsity of the conditional probability distributions. This is expected because of soft functional dependencies that lead the conditioned histograms to possess only a few values.

## 5 Conclusion

The majority of cost models are blindfolded and do not take into account attribute dependencies. This leads to cardinality estimation errors that grow

exponentially and have a negative impact on the query execution time. To prevent this we propose a novel approach based on Bayesian networks to relax the independence assumption. In contrast to prior work also based on Bayesian networks we only capture dependencies inside each relation. This allows our method to be compiled in much less time and to produce selectivity estimates in sub-linear time. We do so by restricting the structure of the network to a tree and by compressing each attribute's conditional probability distributions. We ran our method on a chosen subset of the TPC-DS benchmark and obtained satisfying results. Our method is an order of magnitude more accurate than estimates that assume independence, even though it doesn't attempt to capture cross-relational dependencies. Although our method requires storing a few two-dimensional distributions, the storage requirements are a tiny fraction of those of sampling methods.

Like other table-level synopses, our model does not capture dependencies between attributes of different relations. What's more it doesn't help in determining the size of multi-way joins. In the future we plan to work on these two aspects of the cardinality estimation problem.

## References

1. Acharya, S., Gibbons, P.B., Poosala, V., Ramaswamy, S.: Join synopses for approximate query answering. *ACM SIGMOD Rec.* **28**, 275–286 (1999)
2. Armbrust, M., et al.: Spark SQL: relational data processing in spark. In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pp. 1383–1394. ACM (2015)
3. Bruno, N., Chaudhuri, S., Gravano, L.: STHoles: a multidimensional workload-aware histogram. *ACM SIGMOD Rec.* **30**, 211–222 (2001)
4. Chaudhuri, S., Motwani, R., Narasayya, V.: On random sampling over joins. *ACM SIGMOD Rec.* **28**, 263–274 (1999)
5. Chen, C.M., Roussopoulos, N.: Adaptive selectivity estimation using query feedback, vol. 23. ACM (1994)
6. Chen, Y., Yi, K.: Two-level sampling for join size estimation. In: *Proceedings of the 2017 ACM International Conference on Management of Data*, pp. 759–774. ACM (2017)
7. Chow, C., Liu, C.: Approximating discrete probability distributions with dependence trees. *IEEE Trans. Inf. Theory* **14**(3), 462–467 (1968)
8. Cooper, G.F.: The computational complexity of probabilistic inference using Bayesian belief networks. *Artif. Intell.* **42**(2–3), 393–405 (1990)
9. Cowell, R.G., Dawid, P., Lauritzen, S.L., Spiegelhalter, D.J.: *Probabilistic Networks and Expert Systems: Exact Computational Methods for Bayesian Networks*. Springer Science & Business Media (2006)
10. Getoor, L., Taskar, B., Koller, D.: Selectivity estimation using probabilistic models. *ACM SIGMOD Rec.* **30**, 461–472 (2001)
11. Heckerman, D., Geiger, D., Chickering, D.M.: Learning Bayesian networks: the combination of knowledge and statistical data. *Mach. Learn.* **20**(3), 197–243 (1995)
12. Heimel, M., Kiefer, M., Markl, V.: Self-tuning, GPU-accelerated kernel density models for multidimensional selectivity estimation. In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pp. 1477–1492. ACM (2015)

13. Hellerstein, J.M.: Looking back at postgres. arXiv preprint [arXiv:1901.01973](https://arxiv.org/abs/1901.01973) (2019)
14. Hwang, F.K., Richards, D.S., Winter, P.: The Steiner Tree Problem, vol. 53. Elsevier, Amsterdam (1992)
15. Ioannidis, Y.E., Christodoulakis, S.: On the propagation of errors in the size of join results, vol. 20. ACM (1991)
16. Ioannidis, Y.E., Christodoulakis, S.: Optimal histograms for limiting worst-case error propagation in the size of join results. *ACM Trans. Database Syst. (TODS)* **18**(4), 709–748 (1993)
17. Jaakkola, T., Sontag, D., Globerson, A., Meila, M.: Learning Bayesian network structure using LP relaxations. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 358–365 (2010)
18. Jensen, F.V.: *An Introduction to Bayesian Networks*, vol. 210. UCL Press, London (1996)
19. Kipf, A., Kipf, T., Radke, B., Leis, V., Boncz, P., Kemper, A.: Learned cardinalities: estimating correlated joins with deep learning. arXiv preprint [arXiv:1809.00677](https://arxiv.org/abs/1809.00677) (2018)
20. Kooi, R.P.: *The optimization of queries in relational databases* (1980)
21. Leis, V., Gubichev, A., Mirchev, A., Boncz, P., Kemper, A., Neumann, T.: How good are query optimizers, really? *Proc. VLDB Endowment* **9**(3), 204–215 (2015)
22. Leis, V., et al.: Query optimization through the looking glass, and what we found running the join order benchmark. *VLDB J.* **27**, 1–26 (2018)
23. Leis, V., Radke, B., Gubichev, A., Kemper, A., Neumann, T.: Cardinality estimation done right: index-based join sampling. In: *CIDR* (2017)
24. Li, F., Wu, B., Yi, K., Zhao, Z.: Wander join: online aggregation via random walks. In: *Proceedings of the 2016 International Conference on Management of Data*, pp. 615–629. ACM (2016)
25. Lipton, R.J., Naughton, J.F.: Query size estimation by adaptive sampling. In: *Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp. 40–46. ACM (1990)
26. Moerkotte, G., Neumann, T., Steidl, G.: Preventing bad plans by bounding the impact of cardinality estimation errors. *Proc. VLDB Endowment* **2**(1), 982–993 (2009)
27. Muralikrishna, M., DeWitt, D.J.: Equi-depth multidimensional histograms. *SIGMOD Rec.* **17**, 28–36 (1988)
28. Muthukrishnan, S., Poosala, V., Suel, T.: On rectangular partitionings in two dimensions: algorithms, complexity and applications. In: Beeri, C., Buneman, P. (eds.) *ICDT 1999*. LNCS, vol. 1540, pp. 236–256. Springer, Heidelberg (1999). [https://doi.org/10.1007/3-540-49257-7\\_16](https://doi.org/10.1007/3-540-49257-7_16)
29. Olken, F.: *Random sampling from databases*. Ph.D. thesis, University of California, Berkeley (1993)
30. Piatetsky-Shapiro, G., Connell, C.: Accurate estimation of the number of tuples satisfying a condition. *ACM SIGMOD Rec.* **14**(2), 256–276 (1984)
31. Poess, M., Nambiar, R.O., Walrath, D.: Why you should run TPC-DS: a workload analysis. In: *Proceedings of the 33rd International Conference on Very Large Databases*, pp. 1138–1149. VLDB Endowment (2007)
32. Poosala, V., Haas, P.J., Ioannidis, Y.E., Shekita, E.J.: Improved histograms for selectivity estimation of range predicates. *ACM Sigmod Rec.* **25**, 294–305 (1996)
33. Robertson, N., Seymour, P.D.: Graph minors. II: algorithmic aspects of tree-width. *J. Algorithms* **7**(3), 309–322 (1986)

34. Selinger, P.G., Astrahan, M.M., Chamberlin, D.D., Lorie, R.A., Price, T.G.: Access path selection in a relational database management system. In: Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data, pp. 23–34. ACM (1979)
35. Stillger, M., Lohman, G.M., Markl, V., Kandil, M.: Leo-db2’s learning optimizer. *VLDB* **1**, 19–28 (2001)
36. Traverso, M.: Presto: interacting with petabytes of data at Facebook. Retrieved February 4, 2014 (2013)
37. Tzoumas, K., Deshpande, A., Jensen, C.S.: Lightweight graphical models for selectivity estimation without independence assumptions. *Proc. VLDB Endowment* **4**(11), 852–863 (2011)
38. Vengerov, D., Menck, A.C., Zait, M., Chakkappen, S.P.: Join size estimation subject to filter conditions. *Proc. VLDB Endowment* **8**(12), 1530–1541 (2015)





# An Exploration of Cross-Modal Retrieval for Unseen Concepts

Fangming Zhong<sup>1(✉)</sup>, Zhikui Chen<sup>1,2</sup>, and Geyong Min<sup>3</sup>

<sup>1</sup> School of Software, Dalian University of Technology, Dalian, China  
fmzhong@mail.dlut.edu.cn

<sup>2</sup> Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, Dalian, China

<sup>3</sup> College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter, UK

**Abstract.** Cross-modal hashing has drawn increasing research interests in cross-modal retrieval due to the explosive growth of multimedia big data. However, most of the existing models are trained and tested in a close-set circumstance, which may easily fail on the newly emerged concepts that are never present in the training stage. In this paper, we propose a novel cross-modal hashing model, named Cross-Modal Attribute Hashing (CMAH), which can handle cross-modal retrieval of unseen categories. Inspired by zero-shot learning, attribute space is employed to transfer knowledge from seen categories to unseen categories. Specifically, the cross-modal hashing functions learning and knowledge transfer are conducted by modeling the relationships among features, attributes, and classes as a dual multi-layer network. In addition, graph regularization and binary constraints are imposed to preserve the local structure information in each modality and to reduce quantization loss, respectively. Extensive experiments are carried out on three datasets, and the results demonstrate the effectiveness of CMAH in handling cross-modal retrieval for both seen and unseen concepts.

**Keywords:** Cross-modal retrieval · Unseen classes · Zero-shot learning

## 1 Introduction

Recent years have witnessed the rapidly increasing interests in cross-modal retrieval that is becoming significant and imperative for many real-world applications, such as using image to search the relevant text documents or searching relevant images with given text query [2, 6, 15, 31, 32]. Due to the large-scale and high-dimensional properties of multimodal data, cross-modal hashing which has shown fairly impressive performance in reducing storage cost and improving retrieval speed, has been investigated intensively over the last few years [5, 11, 14, 23, 33].

---

The original version of this chapter was revised: An acknowledgement has been added. The correction to this chapter is available at [https://doi.org/10.1007/978-3-030-18579-4\\_46](https://doi.org/10.1007/978-3-030-18579-4_46)

It is worth noting that, most of the current cross-modal hashing models are trained and tested in a close set i.e. the training and test categories are identical. However, with the explosion of newly-emerging concepts, it is infeasible to label data for each class. Additionally, the number of labelled data for these new concepts may be far from sufficient to build high-quality cross-modal hashing model. The existing methods perform well on the seen data, but they may easily fail on the unseen concepts that are never present before in the training stage. This gives rise to an emerging demand to explore the problem of cross-modal retrieval for unseen concepts.

Such learning with no data in unimodal scenario is termed zero-shot learning (ZSL) [12, 13, 16, 26] that has been widely studied in recent years. The fundamental goal of zero-shot learning is recognizing objects from classes that are not seen during training. The key challenge of achieving this goal is to transfer knowledge from the limited seen categories to unseen categories. Most of the previous approaches employ an intermediate semantic space to conduct knowledge transfer as well as to bridge the semantic gap between low-level visual feature and high-level class label. For example, the authors in [26] proposed to learn semantic embedding projection by matrix tri-factorization and manifold regularization. In [13], semantic autoencoder is employed to learn a projection that can generalize better to new unseen classes. In contrast, orthogonal semantic-visual embedding was developed in [16] to inversely use semantic space to infer visual features for unseen classes. However, all the above methods focus only on unimodal classification or recognition scenarios. Only a few works on zero-shot hashing have been proposed. Zero-shot hashing [29] is one of the first works that focus on handling visual indexing by hashing for unseen categories. In [27], a multi-layer hierarchy was proposed for zero-shot image retrieval. However, in real world, users may be not satisfied with image query, but more comfortable to use other types of query such as text and sound. To the best of our knowledge, the zero-shot learning problem in cross-modal retrieval has been rarely investigated in previous works. In [4], the cross-modal retrieval for unseen categories was first explored with external knowledge. It utilizes a weight vector to build the connection between seen and unseen classes for knowledge transfer. However, this method which simply combines the deep networks with dot product operation uses the pre-trained model with ImageNet that actually includes the information of unseen categories. Thus, the experimental results of cross-modal retrieval for unseen classes are not convincing. Ji et al. also noticed the importance of cross-modal retrieval for unseen concepts [10]. However, their work explores the zero-shot cross-modal hashing with images and the class names, which is different from the traditional cross-modal tasks i.e. image to text and text to image. Therefore, this is the first work that explores cross-modal retrieval for unseen concepts using hashing technique.

In this paper, we consider the problem of handling unseen classes in cross-modal hashing. The main focus is on generalizing the cross-modal hashing model from seen classes to unseen classes, which can produce effective hash codes for data from unseen classes. Motivated by zero-shot hashing and the recently proposed

approaches [21, 27, 29], a novel cross-modal attribute hashing (CMAH) model is presented in this work. During the cross-modal hashing functions learning, the knowledge transfer between seen and unseen categories is conducted with the idea of modelling the relationships among features, attributes, and classes as a dual multi-layer network. As shown in Fig. 1, cross-modal data are projected into unified binary codes that are used to construct the relationship between attributes and class labels, as well as build the connection of different modalities. Moreover, graph regularization and binary constraints are imposed to preserve the local structure information in each modality and to reduce quantization loss, respectively. Thus, the learned hashing functions for each modality through seen classes not only can generate discriminative binary codes for seen classes, but also can generalize well to the unseen classes. By conducting experiments on three non-overlapping cross-modal datasets, the effectiveness of our method has been validated. Compared against the existing cross-modal hashing methods, our method can effectively handle the cross-modal retrieval for unseen concepts. In addition, the proposed method also shows superior performance on the cross-modal retrieval of seen classes.

The main contributions of this paper are:

- A novel cross-modal attribute hashing model is proposed to explore the problem of cross-modal retrieval in zero-shot scenario. To our best knowledge, this is one of the first works which explores the cross-modal hashing for handling unseen classes.
- A cross-modal multi-layer network is developed for simultaneously connecting features, binary codes, attributes, and classes and building relationship among different modalities. Furthermore, local structure information in each modality has been preserved in the expected Hamming space.
- Experiments on cross-modal retrieval for both unseen query and seen query are conducted to evaluate the effectiveness of the proposed method. We find that the proposed method shows superior performance on both the seen query and unseen query.

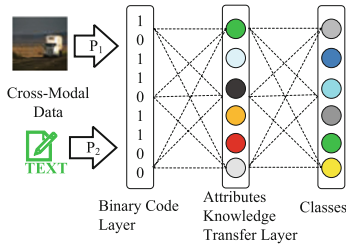
The rest of this paper is organized as follows. The previous works on cross-modal hashing and zero-shot learning are reviewed in Sect. 2. The proposed approach is presented in Sect. 3. Section 4 presents the experimental results. Finally, we conclude our work in Sect. 5.

## 2 Related Work

Since our work mainly concerns handling the unseen classes in cross-modal retrieval based on hashing, this section reviews the previous works from two aspects, i.e. conventional cross-modal hashing and unimodal zero-shot learning.

### 2.1 Cross-Modal Hashing

Motivated by hashing, a number of methods have been proposed to conduct cross-modal retrieval. For example, in Collective Matrix Factorization Hashing



**Fig. 1.** The framework of the proposed method.

(CMFH) [5], matrix factorization is utilized to learn the latent concepts from each modality, which has achieved an impressive result on cross-modal retrieval. In [33], Latent Semantic Sparse Hashing (LSSH) was presented which learns the semantic concepts of images and text by sparse coding and matrix factorization respectively. The learned latent semantic features from images and text are then mapped to a common abstraction space in which the unified hash codes are generated by quantization. Inspired by CMFH, several supervised extensions [14, 23] have been proposed to formulate the label information for boosting retrieval performance. A unified linear regression model with dragging technique based on semi-supervised learning for cross-modal retrieval was proposed in [30]. Most of these methods adopt hand-crafted features as input. Recently, deep neural networks such as convolutional neural networks (CNN) have drawn considerable attention in cross-modal retrieval [11, 28]. Due to the high-level abstract of original data, the CNN based methods perform better than those based on low-level hand-crafted features. However, these methods suffer from high time complexity of training CNN. Most importantly, none of hand-crafted or CNN based methods have considered the cross-modal retrieval for unseen concepts.

## 2.2 Zero-Shot Learning

Zero-shot learning has become an active topic in recent years due to the rapid evolution of newly-emerging concepts. Most of the existing methods aim at solving the recognition task of unseen categories. A promising solution is to find an intermediate representation which can bridge the semantic gap between visual features and class labels, and can also transfer knowledge from seen classes to unseen classes. For instance, the methods including [12, 16] project both the images and class labels into an attribute space, where a simple nearest neighbor classifier can be adopted to recognize the instances from unseen categories. However, these methods focus only on classification or recognition. Few works on zero-shot hashing for retrieval have been presented. In [18], the authors investigated the hashing in the zero shot scenario for image retrieval, in which hashing function is learned based on the combination of similarity preserving and unsupervised domain adaptation. In [8], a zero-shot hashing based on CNN is proposed, which considers the similarity transfer, discriminability, and discrete

hashing comprehensively. However, the existing zero-shot hashing approaches can only deal with single modality, the circumstance of multiple modalities has not been explored. Therefore, a novel cross-modal hashing framework that can handle cross-modal retrieval for unseen classes draws a significant need.

Overall, the cross-modal retrieval for unseen concepts has not yet been investigated well. The works in [4] and [10] have noticed the importance of this problem. However, due to the utilization of pre-trained model for feature extraction and retrieval based on class name, their results are not convincing enough. To our best knowledge, this paper is one of the first works that explore cross-modal retrieval for unseen concepts using hashing technique.

### 3 Approach

In this section, the proposed CMAH for tackling cross-media retrieval of unseen classes is described in detail followed by the optimization algorithm.

#### 3.1 Problem Definition

The definition of zero-shot cross-modal hashing follows [29]. Given  $n$  pairs of “seen” cross-modal data  $\mathbf{X}^{(1)} = \{x_1^{(1)}, \dots, x_n^{(1)}\}$  and  $\mathbf{X}^{(2)} = \{x_1^{(2)}, \dots, x_n^{(2)}\}$ , such as images and the associated text, where  $\mathbf{X}^{(1)} \in \mathbb{R}^{d_1 \times n}$ ,  $\mathbf{X}^{(2)} \in \mathbb{R}^{d_2 \times n}$ ,  $d_1$  represents the dimensionality of image feature,  $d_2$  denotes the dimensionality of text feature (usually  $d_1 \neq d_2$ ). The semantic label of the given data is  $\mathbf{Y} \in \{0, 1\}^{n \times c}$ , where  $c$  is the size of “seen” classes. Different from the conventional cross-modal hashing setting in which the training data and testing data are from the seen classes, we assume that some testing data are from unseen classes which are never present during training. The goal of our proposed method is to learn cross-modal hashing model via seen classes and then generalize it to unseen classes for generating high-quality discriminative binary codes.

#### 3.2 Cross-Modal Attribute Hashing Formulation

Motivated by the recently proposed zero-shot learning approaches [21], the knowledge transfer is conducted in the intermediate attribute space. As shown in Fig. 1, we formulate the relationship of cross-modal data, binary codes, attribute, and class labels as the following loss function:

$$\begin{aligned} \mathcal{L}_1 = & \|\mathbf{Y} - \mathbf{B}\mathbf{V}\mathbf{S}\|_F^2 + \alpha \left\| \mathbf{B} - (\mathbf{X}^{(1)})^T \mathbf{P}_1 \right\|_F^2 \\ & + \beta \left\| \mathbf{B} - (\mathbf{X}^{(2)})^T \mathbf{P}_2 \right\|_F^2 \\ \text{s.t. } & \mathbf{B} \in \{-1, +1\}^{n \times k} \end{aligned} \quad (1)$$

where  $\mathbf{B}$  is the unified binary codes of  $\mathbf{X}^{(1)}$  and  $\mathbf{X}^{(2)}$ ,  $k$  is the length of binary codes,  $\mathbf{P}_1 \in \mathbb{R}^{d_1 \times k}$  and  $\mathbf{P}_2 \in \mathbb{R}^{d_2 \times k}$  are cross-modal hashing functions projecting

images and text to hash codes, and  $\mathbf{V} \in \mathbb{R}^{k \times a}$  is the mapping matrix from binary codes to attributes,  $\mathbf{S} \in \mathbb{R}^{a \times c}$  is the mapping from attributes to semantic class labels, where  $a$  is the number of attributes. Here, we use the word vector representation of class name as  $\mathbf{S}$  following the idea of [3].

In addition, the local structure information preserving in each modality is explored during learning cross-modal hashing functions. Thus, the similar items in original feature space will share similar binary codes in the Hamming space, which can further enhance the discriminative capability of learned hashing functions. Laplacian Eigenmaps (LE) [1] is utilized to formulate the structure preservation based on manipulations on an undirected weight graph which indicates the neighborhood relationship of pairwise data. The objective with respect to  $\mathbf{X}^{(1)}$  can be stated as:

$$\min_{\mathbf{P}_1} \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \left\| \mathbf{P}_1^T x_i^{(1)} - \mathbf{P}_1^T x_j^{(1)} \right\|^2 w_{ij}^{(1)} \quad (2)$$

where  $w_{ij}^{(1)}$  is the similarity of  $x_i^{(1)}$  and  $x_j^{(1)}$ . It usually can be defined according to the neighborhood relationship as below:

$$w_{ij}^{(1)} = \begin{cases} \exp\left(-\frac{\|x_i^{(1)} - x_j^{(1)}\|^2}{2\sigma^2}\right), & \text{if } x_i^{(1)} \in \mathcal{N}_k(x_j^{(1)}) \text{ or } x_j^{(1)} \in \mathcal{N}_k(x_i^{(1)}), \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

where  $\mathcal{N}_k(x_j^{(1)})$  is the  $k$ -nearest neighbors of  $x_j^{(1)}$ . The Euclidean distance between samples  $x_i^{(1)}$  and  $x_j^{(1)}$  is used for finding nearest neighbors.  $\sigma$  is the bandwidth parameter which is set to  $\sigma = 1$  in our experiments.

Through algebraic calculation, the objective function in Eq. (2) can be reformulated as:

$$\min_{\mathbf{P}_1} \text{tr}(\mathbf{P}_1^T \mathbf{X}^{(1)} \mathbf{L}_1 (\mathbf{X}^{(1)})^T \mathbf{P}_1) \quad (4)$$

where  $\mathbf{L}_1$  is the Laplacian matrix,  $\mathbf{L}_1 = \mathbf{D}_1 - \mathbf{W}^{(1)}$ ,  $\mathbf{D}_1$  is a diagonal matrix,  $\mathbf{D}_1(i, i) = \sum_j w_{ij}^{(1)}$ . The elements of  $\mathbf{W}^{(1)}$  are  $w_{ij}^{(1)}$ .  $\text{tr}(\cdot)$  is the trace operator. Similarly, for modality  $\mathbf{X}^{(2)}$ , we can have:

$$\min_{\mathbf{P}_2} \text{tr}(\mathbf{P}_2^T \mathbf{X}^{(2)} \mathbf{L}_2 (\mathbf{X}^{(2)})^T \mathbf{P}_2) \quad (5)$$

where  $\mathbf{L}_2$  is the Laplacian matrix of  $\mathbf{X}^{(2)}$ . Finally, combining the relationship modeling and local structure information preserving, the overall objective can be stated as follows:

$$\begin{aligned} & \min_{\mathbf{B}, \mathbf{V}, \mathbf{P}_1, \mathbf{P}_2} \mathcal{L}_1 + \mathcal{L}_2 + \Omega(\mathbf{B}, \mathbf{V}, \mathbf{S}, \mathbf{P}_1, \mathbf{P}_2) \\ & \text{s.t. } \mathbf{B} \in \{-1, +1\}^{n \times k} \end{aligned} \quad (6)$$

where

$$\mathcal{L}_2 = \lambda_1 \text{tr}(\mathbf{P}_1^T \mathbf{X}^{(1)} \mathbf{L}_1 (\mathbf{X}^{(1)})^T \mathbf{P}_1) + \lambda_2 \text{tr}(\mathbf{P}_2^T \mathbf{X}^{(2)} \mathbf{L}_2 (\mathbf{X}^{(2)})^T \mathbf{P}_2) \quad (7)$$

where  $\lambda_1, \lambda_2$  are balancing parameters. Inspired by [21], a regularization term is also integrated which is defined as:

$$\begin{aligned} \Omega(\mathbf{B}, \mathbf{V}, \mathbf{S}, \mathbf{P}_1, \mathbf{P}_2) \\ = \gamma \|\mathbf{VS}\|_F^2 + \mu \|\mathbf{BS}\|_F^2 + \gamma\mu \|\mathbf{V}\|_F^2 + \xi_1 \|\mathbf{P}_1\|_F^2 + \xi_2 \|\mathbf{P}_2\|_F^2 \end{aligned} \quad (8)$$

where  $\gamma, \mu, \xi_1$ , and  $\xi_2$  are trade-off parameters.

### 3.3 Optimization

It is intractable to directly minimize the objective in Eq. (6) because of the non-convexity with four matrix variables  $\mathbf{P}_1, \mathbf{P}_2, \mathbf{V}$ , and  $\mathbf{B}$ . Fortunately, it is convex with respect to any of them when the others are fixed. Therefore, we employ an alternative optimization in an iterative manner to address the optimization problem until convergence. The detailed optimization steps are listed as follows:

**Update  $\mathbf{P}_1, \mathbf{P}_2$ .** Fix other variables but  $\mathbf{P}_1$ , then the objective function shown in Eq. (6) can be simplified as:

$$\begin{aligned} \min_{\mathbf{P}_1} \lambda_1 \text{tr}(\mathbf{P}_1^T \mathbf{X}^{(1)} \mathbf{L}_1 (\mathbf{X}^{(1)})^T \mathbf{P}_1) \\ + \alpha \left\| \mathbf{B} - (\mathbf{X}^{(1)})^T \mathbf{P}_1 \right\|_F^2 + \xi_1 \|\mathbf{P}_1\|_F^2 \end{aligned} \quad (9)$$

By setting its derivative w.r.t  $\mathbf{P}_1$  to 0, we can have the closed-form solution stated as follows:

$$\mathbf{P}_1 = (\alpha \mathbf{X}^{(1)} (\mathbf{X}^{(1)})^T + \lambda_1 \mathbf{X}^{(1)} \mathbf{L}_1 (\mathbf{X}^{(1)})^T + \xi_1 \mathbf{I})^{-1} \alpha \mathbf{X}^{(1)} \mathbf{B} \quad (10)$$

Similarly,  $\mathbf{P}_2$  can be updated by:

$$\mathbf{P}_2 = (\beta \mathbf{X}^{(2)} (\mathbf{X}^{(2)})^T + \lambda_2 \mathbf{X}^{(2)} \mathbf{L}_2 (\mathbf{X}^{(2)})^T + \xi_2 \mathbf{I})^{-1} \beta \mathbf{X}^{(2)} \mathbf{B} \quad (11)$$

**Update  $\mathbf{V}$ .** The objective can be transformed to the following when fixing the other variables but  $\mathbf{V}$ :

$$\min_{\mathbf{V}} \|\mathbf{Y} - \mathbf{BVS}\|_F^2 + \gamma \|\mathbf{VS}\|_F^2 + \mu \|\mathbf{BV}\|_F^2 + \gamma\mu \|\mathbf{V}\|_F^2 \quad (12)$$

By setting its derivative w.r.t  $\mathbf{V}$  to 0, we can have the closed-form solution stated as follows:

$$\mathbf{V} = (\mathbf{B}^T \mathbf{B} + \gamma \mathbf{I})^{-1} \mathbf{B}^T \mathbf{YS}^T (\mathbf{SS}^T + \mu \mathbf{I})^{-1} \quad (13)$$

**Update  $\mathbf{B}$ .** Fixing other variables but  $\mathbf{B}$ , we can learn the unified binary codes of image and text directly without relaxation by solving the reformulated optimization stated as:

$$\begin{aligned} \min_{\mathbf{B}} \|\mathbf{Y} - \mathbf{BVS}\|_F^2 + \alpha \left\| \mathbf{B} - (\mathbf{X}^{(1)})^T \mathbf{P}_1 \right\|_F^2 \\ + \beta \left\| \mathbf{B} - (\mathbf{X}^{(2)})^T \mathbf{P}_2 \right\|_F^2 + \mu \|\mathbf{BV}\|_F^2 \\ \text{s.t. } \mathbf{B} \in \{-1, +1\}^{n \times k} \end{aligned} \quad (14)$$

The optimization defined in Eq. (14) under binary constraint can be easily solved by using discrete cyclic coordinate descent (DCC) method [22].

The proposed model is summarized in Algorithm 1. Through alternative optimization, the objective is minimized in each iterative step, and it will converge in the end.

## 4 Experiments

Extensive experiments are carried out in this section to evaluate the effectiveness of the proposed method in cross-modal retrieval, where some classes may not have been seen during training. Two tasks i.e. text to image (T2I) and image to text (I2T), are designed to validate the proposed approach in handling “seen” and “unseen” cross-modal retrieval. In our experiments, an image and a text are considered to be relevant if they share the same semantic label.

### 4.1 Datasets

**Wiki** [20] dataset consists of 2866 image-text documents. These documents can be grouped into 10 semantic categories. The images are described in 128-dimensional bag-of-visual words SIFT feature vectors, while text is represented by 10-dimensional topic vectors generated by the latent Dirichlet allocation (LDA) model.

**Pascal VOC** [7] dataset contains 9963 testing image-tag pairs, which can be classified into 20 categories. Since several image-tag pairs are multi-labeled, we select the pairs with only one label as the way in [24]. The image modality is represented by 512-dimensional GIST features [9], and the representations of text modality are 399-dimensional word frequency features.

**LabelMe** [17] dataset contains 2688 outdoor scenes from 8 different classes. We discard the words that occur in less than 3 times, resulting in 366 unique words. Thus, the representation of text is a 366-dimensional word frequency. The images are represented by 512-dimensional GIST features. Additionally, we delete the samples without tags, which results in a dataset with 2686 image-text pairs.

All the datasets are completely mutually exclusive, i.e. no overlapping samples between classes. In terms of attribute mapping  $\mathbf{S}$ , the word vectors of class names extracted from GloVe [19] are used in our experiments.

### 4.2 Settings

We construct the zero-shot scenario for three datasets as follows. For Wiki and LabelMe, we randomly select 2 classes as the unseen concepts each time, and the rest as seen classes. For Pascal dataset 4 classes are randomly selected as unseen classes each time. We report the average result of 10 experiments with randomly selected unseen concepts.

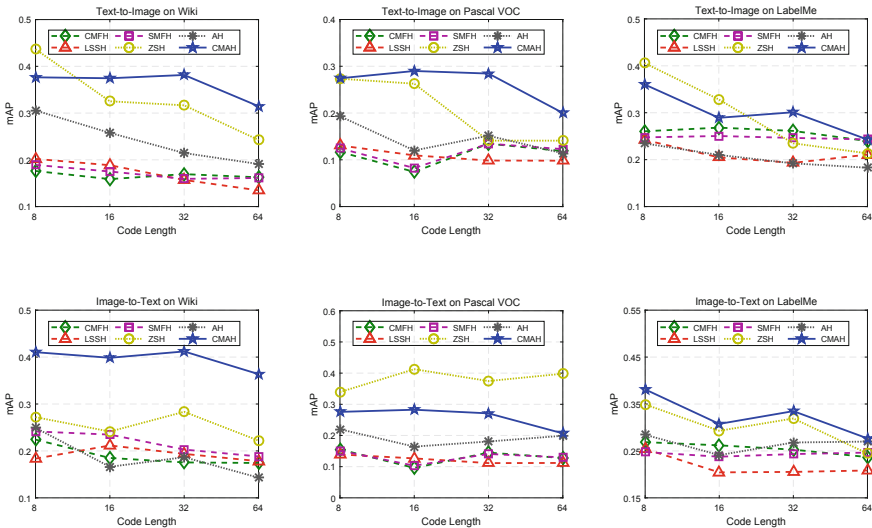


**Algorithm 1.** CMAH

**Input:** Seen cross-modal data  $\mathbf{X}^{(1)}$  and  $\mathbf{X}^{(2)}$ , label  $\mathbf{Y}$ , attribute mapping  $\mathbf{S}$ , the length of hash codes  $k$ , and parameters  $\alpha, \beta, \lambda_1, \lambda_2, \xi_1, \xi_2, \mu$ , and  $\gamma$ .

**Output:** Unified hash codes  $\mathbf{B}$ , hashing functions  $\mathbf{P}_1, \mathbf{P}_2$ .

- 1: Compute Laplacian matrix  $\mathbf{L}_1, \mathbf{L}_2$
- 2: Initialize  $\mathbf{P}_1, \mathbf{P}_2, \mathbf{B}, \mathbf{V}$ .
- 3: **repeat**
- 4:   Update  $\mathbf{P}_1, \mathbf{P}_2$  by Eqs. (10) and (11);
- 5:   Update  $\mathbf{V}$  by Eq. (13);
- 6:   Update  $\mathbf{B}$  by solving Eq. (14);
- 7: **until** convergence
- 8: **return**  $\mathbf{B}, \mathbf{P}_1, \mathbf{P}_2$



**Fig. 2.** MAP on unseen query of various approaches with varied hash code lengths on three datasets.

In order to evaluate the performance on handling unseen classes cross-modal retrieval. We use the testing data of unseen classes as query set, and we construct the retrieval set by merging the retrieval set of both seen and unseen classes, which follows the generalized zero-shot setting in [25]. In addition, the proposed new model should still have the comparable or even better performance on the seen classes. To this end, extra experiments of cross-modal retrieval are designed on the seen classes. The testing data of seen classes are chosen as query set, and the training data are regarded as retrieval set.

Two widely used metrics are employed to evaluate the performance of cross-modal retrieval. One is the mean Average Precision (mAP) based on Hamming ranking of all the retrieval set. The other is the mean precision within Hamming distance radius 2 (PH2) based on lookup table.

In our experiments, we empirically set  $\alpha$  and  $\beta$  to 0.1. For balancing parameters  $\lambda_1$  and  $\lambda_2$ , we set them to 0.1. The trade-off parameters  $\gamma$ ,  $\mu$ ,  $\xi_1$ , and  $\xi_2$  are set to  $10^{-3}$ ,  $10^{-3}$ , 10, and 10, respectively. The Laplacian matrix is constructed within the 5 nearest neighbors. For the optimization procedure, we restrain the iteration number to 10.

### 4.3 Baselines

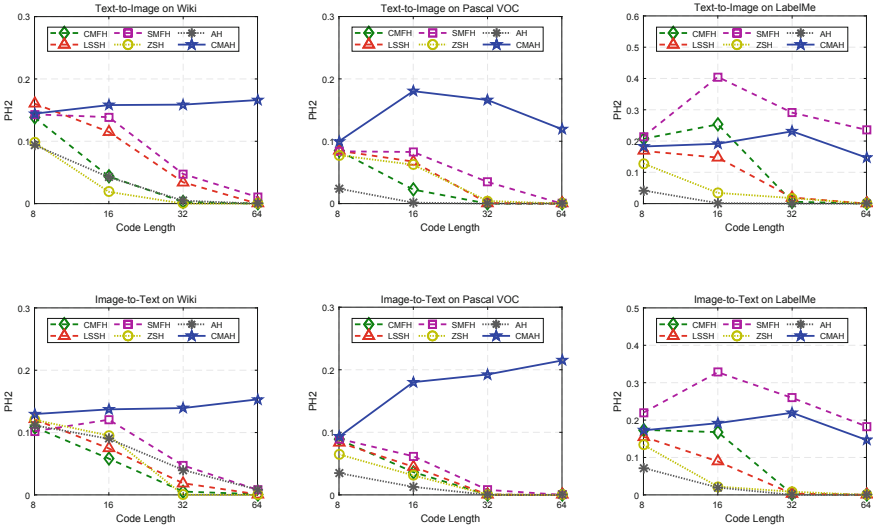
Since this is the first work of cross-modal hashing which considers the zero-shot scenario, we compare the proposed CMAH against five state-of-the-art methods including three conventional cross-modal hashing methods and two zero-shot hashing methods. The former three are CMFH [5], LSSH [33], and Supervised Matrix Factorization Hashing (SMFH) [23], respectively. The latter two are Attribute Hashing (AH) [27] and Zero-Shot Hashing (ZSH) [29]. In particular, for unimodal methods AH and ZSH, we only fix the length of hash codes such as 8 bits. Then hashing functions are trained independently on image and text modality. In the testing phase, the binary codes of text and image for query and retrieval are generated by the learned hashing functions respectively. The parameters of the baselines are set according to the suggestion of their original papers.

### 4.4 Experimental Results

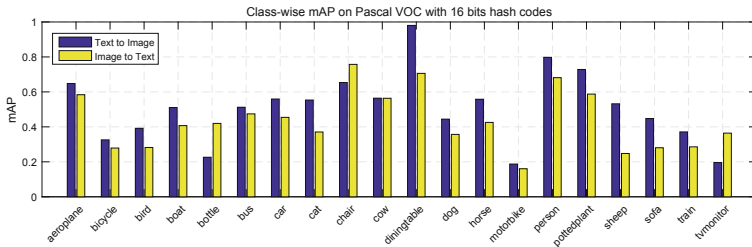
**Results on Unseen Query.** Firstly, the performance of handling cross-modal retrieval for unseen concepts is evaluated from two aspects.

First, the mAP results that indicate the overall performance of cross-modal retrieval for unseen query are shown in Fig. 2. Compared against the conventional cross-modal hashing approaches CMFH, LSSH, and SMFH, the proposed method CMAH outperforms them with significant margins in most cases. This is because they are trained in a close-set circumstance, which makes it limited to generalize the hashing functions to newly emerged concepts that have never been present. We also notice that AH and ZSH perform better than CMFH, LSSH, and SMFH. Since they are excellent zero-shot learning methods, they can handle the knowledge transfer from seen to unseen classes. However, AH and ZSH are unimodal methods that ignore the correlation of different modalities. Thus, our CMAH outperforms AH and ZSH in most cases. It demonstrates the effectiveness of our proposed CMAH in handling cross-modal retrieval for unseen concepts.

Second, the precision within Hamming radius 2 (PH2) that indicates the local distribution performance which reveals how far the relevant instances is from the query item is plotted in Fig. 3. It can be seen that our method outperforms others in most cases except SMFH on the Labelme dataset. This outlier may be caused by the weak correlations between unseen and seen classes in LabelMe. Different from the mAP results, the PH2 of AH and ZSH are inferior than CMFH, LSSH, and SMFH. This is because the encoding of cross-modal correlation enhances



**Fig. 3.** Precision on unseen query of various approaches with varied hash code lengths on three datasets.



**Fig. 4.** Class-wise mAP of unseen query on Pascal VOC with 16 bits hash codes.

the performance of cross-modal methods, which is why the unimodal zero-shot methods AH and ZSH obtain higher mAP but lower precision in most cases.

Moreover, from Fig. 2, we can find that all the methods show a slightly trend toward degradation. The trend of PH2 of all baselines is similar to mAP but with more rapid decreasing speed. It is because the longer binary codes can carry more discriminant information but also introduces noise into the codes. In contrast, our method generally has a rising trend of PH2 from 8 bits to 32 bits. It demonstrates that the overall performance decreases slightly as the code length increases, but more relevant instances are distributed around the query item. It further depicts our cross-modal multi-layer network can enhance the robustness to noise. The proposed CMAH is thus able to generate hash codes for unseen query with high discriminative capability.

An additional observation to the hash code length is that as it increases, the mAP shows a slightly trend toward degradation, while the precision has a

rising trend from 8 bits to 16 bits and then decreases rapidly from 16 bits to 64 bits. This phenomenon indicates that an appropriate hash code length which can balance the information encoding and noise is significant for cross-modal retrieval for unseen concepts.

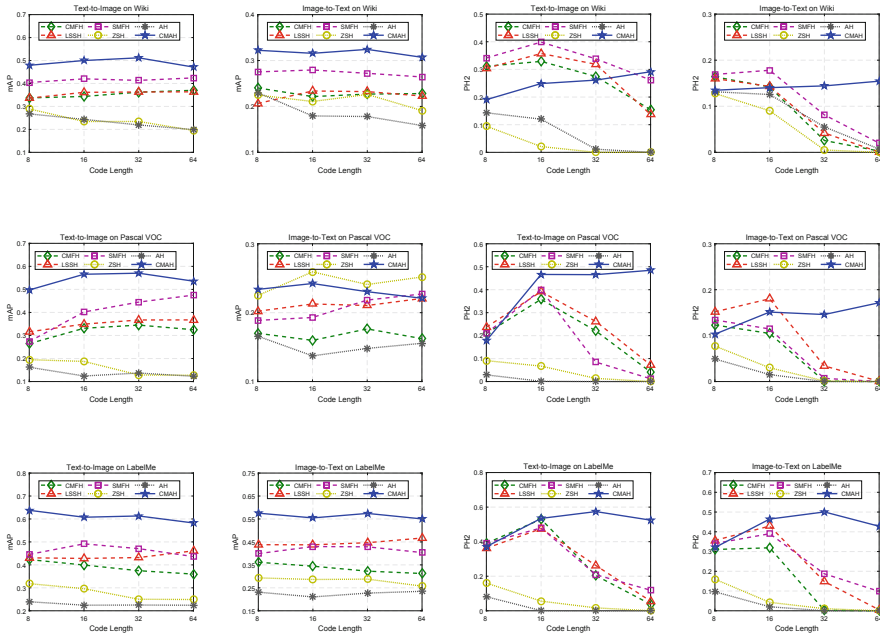
We also investigate the correlation between unseen and seen classes by conducting additional experiments on Pascal VOC. One class is selected as the unseen concept and the rest as seen concepts each time, and thus we have 20 different splits. The mAP results of cross-modal retrieval for the 20 unseen classes are shown in Fig. 4, respectively. Generally, the T2I task performs better than I2T task. More importantly, we find that the unseen class that shares more similar attributes with seen classes will lead to a better performance when conducting unseen query. For example, ‘dining table’ is quit close to four-leg ‘chair’, and ‘cow’ is similar to ‘horse’ and ‘sheep’. Therefore, the selection of unseen classes will affect the performance of cross-modal retrieval for unseen concepts. For this reason, we present the average performance of repeated experiments with randomly selected unseen classes.

**Results on Seen Query.** Then, we still evaluate the performance of our method on seen query i.e. the same test setting with conventional cross-modal retrieval methods. The mAP results of all approaches are reported in Table 1. Similarly, the T2I task outperforms I2T tasks. This is because the representation of text feature is closer to the object semantic than the visual feature. Different from unseen query, it can be seen that AH and ZSH perform rather poorly, while CMFH, LSSH, SMFH perform well in the cross-modal retrieval for seen classes. This is because CMFH, SMFH and LSSH are trained in a close-set circumstance, which makes them limited to generalize the hashing functions to unseen concepts. We can see that the results of our method with varied code lengths are superior to AH and ZSH with large margin. Compared to the best cross-modal hashing method, our proposed CMAH performs comparable or even better on the three datasets. The reason is that our CMAH strives to achieve a balance between unseen and seen query. CMAH mainly focus on the knowledge transfer for unseen classes, which degrades slightly the discriminant of generated binary codes of seen classes. However, due to the utilization of attributes the binary codes can carry additional discrimination information, which results in superior performance in some cases such as on LabelMe and Pascal datasets. In addition, a rising trend is observed on seen query as the code length increases.

**Overall Results.** Finally, we analyze the average result of cross-modal retrieval for seen and unseen classes. The average results on three datasets are plotted in Fig. 5. Generally, it can be observed that our proposed CMAH is superior to the others, which demonstrates the effectiveness of our method. More over, the cross-modal methods CMFH, LSSH, SMFH, and our CMAH perform better than the unimodal method AH and ZSH. In the view of mAP, our method outperforms others in most cases except ZSH on Pascal with Image to Text task. The results of cross-modal methods increase steadily as the code length varies from 8 to

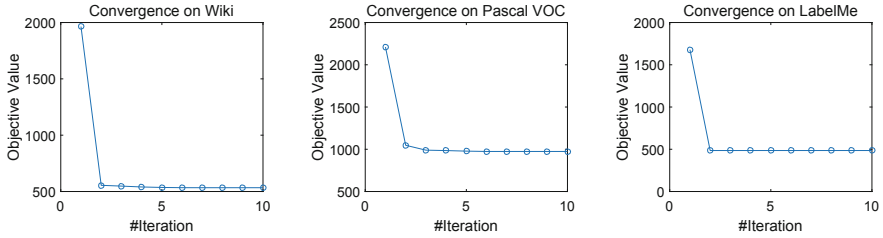
**Table 1.** MAP results on seen query of all methods with varied hash code lengths on three datasets.

Task	Method	Wiki				Pascal VOC				LabelMe			
		8 bits	16 bits	32 bits	64 bits	8 bits	16 bits	32 bits	64 bits	8 bits	16 bits	32 bits	64 bits
T2I	CMFH	0.4982	0.5281	0.5549	0.5779	0.4125	0.5885	0.5558	0.5314	0.5856	0.5306	0.4879	0.4801
	LSSH	0.4698	0.5345	0.5688	0.5926	0.5013	0.5891	0.6362	0.6372	0.6192	0.6513	0.6707	0.7116
	SMFH	<b>0.6195</b>	<b>0.6653</b>	<b>0.6677</b>	<b>0.6874</b>	0.4231	0.7216	0.7553	0.8277	0.6438	0.7329	0.6951	0.6291
	ZSH	0.1418	0.1435	0.1522	0.1484	0.1165	0.1122	0.1161	0.1139	0.2305	0.2643	0.2657	0.2847
	AH	0.2294	0.2295	0.224	0.2074	0.1312	0.1283	0.1226	0.1319	0.2441	0.2386	0.2579	0.2656
	CMAH	0.5819	0.6256	0.6420	0.6282	<b>0.7208</b>	<b>0.8417</b>	<b>0.8564</b>	<b>0.8713</b>	<b>0.9151</b>	<b>0.926</b>	<b>0.9252</b>	<b>0.9222</b>
I2T	CMFH	0.2565	0.2579	0.2783	0.2817	0.1848	0.2235	0.2078	0.1969	0.4540	0.4266	0.3927	0.3879
	LSSH	0.2292	0.2557	0.2708	0.2673	<b>0.2650</b>	<b>0.2987</b>	<b>0.3094</b>	<b>0.3294</b>	0.6217	0.6705	0.6878	0.7256
	SMFH	<b>0.3089</b>	<b>0.3240</b>	<b>0.3414</b>	<b>0.3402</b>	0.2266	0.2828	0.2959	0.3245	0.5504	0.6221	0.6140	0.5607
	ZSH	0.1777	0.1803	0.1689	0.1597	0.1105	0.1059	0.1071	0.1055	0.2376	0.2798	0.2566	0.2704
	AH	0.2107	0.1924	0.1694	0.1736	0.1119	0.1103	0.1155	0.1120	0.1777	0.1796	0.1861	0.1998
	CMAH	0.2342	0.2329	0.2365	0.2503	0.1919	0.2020	0.1900	0.2351	<b>0.7694</b>	<b>0.8017</b>	<b>0.8136</b>	<b>0.8239</b>

**Fig. 5.** Average results of seen and unseen query of various approaches with varied hash code lengths on three datasets.

64 bits. Whereas AH and ZSH present a slightly trend toward degradation. In terms of PH2, all the baselines reach their peaks at 16 bits and then decrease dramatically. The overall performance of our proposed CMAH shows a rising trend on three datasets.

Therefore, we can conclude that our novel model is effective and the competitive in cross-modal retrieval for both seen and unseen concepts.



**Fig. 6.** Convergence analysis.

## 4.5 Convergence Analysis

The convergence of our method is evaluated via empirical experiments on Wiki, Pascal VOC, and LabelMe when hash code length is 16 bits. As shown in Fig. 6, our method can swiftly converge within 5 iterations, which demonstrates its efficiency in real-life applications.

## 5 Conclusion

In this paper, an exploration on the zero-shot problem in cross-modal retrieval is conducted. We proposed a novel cross-modal attribute hashing model that can generalize the hashing functions to newly emerged concepts. A dual multi-layer network is developed, where attribute plays a crucial role in not only helping to well transfer knowledge from seen to unseen concepts, but also narrowing down the semantic gap across visual features, text features, and class labels. Experiments demonstrated the effectiveness of our model in cross-modal retrieval for both seen and unseen concepts. With this initial exploration, many problems are still worthy of further investigation, such as the selection of unseen classes and the balance between seen query and unseen query.

**Acknowledgement.** This work was supported in part by the National Key Research and Development Program of China (2018YFC0831305) and in part by the Nature Science Foundation of China (61672123).

## References

1. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.* **15**(6), 1373–1396 (2003)
2. Cao, Y., Long, M., Wang, J., Liu, S.: Collective deep quantization for efficient cross-modal retrieval. In: *AAAI*, pp. 3974–3980 (2017)
3. Changpinyo, S., Chao, W.L., Gong, B., Sha, F.: Synthesized classifiers for zero-shot learning. In: *CVPR*, pp. 5327–5336 (2016)
4. Chi, J., Huang, X., Peng, Y.: Zero-shot cross-media retrieval with external knowledge. In: Huet, B., Nie, L., Hong, R. (eds.) *ICIMCS 2017. CCIS*, vol. 819, pp. 200–211. Springer, Singapore (2018). [https://doi.org/10.1007/978-981-10-8530-7\\_20](https://doi.org/10.1007/978-981-10-8530-7_20)

5. Ding, G., Guo, Y., Zhou, J.: Collective matrix factorization hashing for multimodal data. In: CVPR, pp. 2075–2082 (2014)
6. Ding, K., Fan, B., Huo, C., Xiang, S., Pan, C.: Cross-modal hashing via rank-order preserving. *IEEE Trans. Multimedia* **19**(3), 571–585 (2017). <https://doi.org/10.1109/TMM.2016.2625747>
7. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The Pascal visual object classes (VOC) challenge. *Int. J. Comput. Vis.* **88**(2), 303–338 (2010)
8. Guo, Y., Ding, G., Han, J., Gao, Y.: SitNet: discrete similarity transfer network for zero-shot hashing. In: IJCAI, pp. 1767–1773 (2017)
9. Hwang, S.J., Grauman, K.: Reading between the lines: object localization using implicit cues from image tags. *IEEE Trans. Pattern Anal. Mach. Intell.* **34**(6), 1145–1158 (2012)
10. Ji, Z., Sun, Y., Yu, Y., Pang, Y., Han, J.: Attribute-guided network for cross-modal zero-shot hashing. arXiv preprint [arXiv:1802.01943](https://arxiv.org/abs/1802.01943) (2018)
11. Jiang, Q.Y., Li, W.J.: Deep cross-modal hashing. In: CVPR, pp. 3270–3278 (2017)
12. Kodirov, E., Xiang, T., Fu, Z., Gong, S.: Unsupervised domain adaptation for zero-shot learning. In: CVPR, pp. 2452–2460 (2015)
13. Kodirov, E., Xiang, T., Gong, S.: Semantic autoencoder for zero-shot learning. In: CVPR, pp. 3174–3183 (2017)
14. Liu, H., Ji, R., Wu, Y., Hua, G.: Supervised matrix factorization for cross-modality hashing. In: IJCAI, pp. 1767–1773 (2016)
15. Liu, L., Lin, Z., Shao, L., Shen, F., Ding, G., Han, J.: Sequential discrete hashing for scalable cross-modality similarity retrieval. *IEEE Trans. Image Process.* **26**(1), 107–118 (2017)
16. Long, Y., Liu, L., Shao, L.: Towards fine-grained open zero-shot learning: inferring unseen visual features from attributes. In: IEEE Winter Conference on Applications of Computer Vision, pp. 944–952 (2017)
17. Oliva, A., Torralba, A.: Modeling the shape of the scene: a holistic representation of the spatial envelope. *Int. J. Comput. Vis.* **42**(3), 145–175 (2001)
18. Pachori, S., Deshpande, A., Raman, S.: Hashing in the zero shot framework with domain adaptation. *Neurocomputing* **275**, 2137–2149 (2018)
19. Pennington, J., Socher, R., Manning, C.: GloVe: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, pp. 1532–1543 (2014)
20. Rasiwasia, N., et al.: A new approach to cross-modal multimedia retrieval. In: Proceedings of the 18th ACM International Conference on Multimedia, pp. 251–260 (2010)
21. Romera-Paredes, B., Torr, P.: An embarrassingly simple approach to zero-shot learning. In: International Conference on Machine Learning, pp. 2152–2161 (2015)
22. Shen, F., Shen, C., Liu, W., Tao Shen, H.: Supervised discrete hashing. In: CVPR, pp. 37–45 (2015)
23. Tang, J., Wang, K., Shao, L.: Supervised matrix factorization hashing for cross-modal retrieval. *IEEE Trans. Image Process.* **25**(7), 3157–3166 (2016)
24. Wang, K., He, R., Wang, L., Wang, W., Tan, T.: Joint feature selection and subspace learning for cross-modal retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.* **38**(10), 2010–2023 (2016)
25. Xian, Y., Schiele, B., Akata, Z.: Zero-shot learning-the good, the bad and the ugly. In: CVPR, pp. 4582–4591 (2017)
26. Xu, X., Shen, F., Yang, Y., Zhang, D., Shen, H.T., Song, J.: Matrix tri-factorization with manifold regularizations for zero-shot learning. In: CVPR (2017)

27. Xu, Y., Yang, Y., Shen, F., Xu, X., Zhou, Y., Shen, H.T.: Attribute hashing for zero-shot image retrieval. In: IEEE International Conference on Multimedia and Expo, pp. 133–138 (2017)
28. Yang, E., Deng, C., Liu, W., Liu, X., Tao, D., Gao, X.: Pairwise relationship guided deep hashing for cross-modal retrieval. In: AAAI, pp. 1618–1625 (2017)
29. Yang, Y., Luo, Y., Chen, W., Shen, F., Shao, J., Shen, H.T.: Zero-shot hashing via transferring supervised knowledge. In: Proceedings of the 2016 ACM on Multimedia Conference, pp. 1286–1295 (2016)
30. Zhang, L., Ma, B., He, J., Li, G., Huang, Q., Tian, Q.: Adaptively unified semi-supervised learning for cross-modal retrieval. In: AAAI, pp. 3406–3412 (2017)
31. Zhang, L., Ma, B., Li, G., Huang, Q., Tian, Q.: Generalized semi-supervised and structured subspace learning for cross-modal retrieval. *IEEE Trans. Multimedia* **20**(1), 128–141 (2018)
32. Zhong, F., Chen, Z., Min, G.: Deep discrete cross-modal hashing for cross-media retrieval. *Pattern Recogn.* **83**, 64–77 (2018)
33. Zhou, J., Ding, G., Guo, Y.: Latent semantic sparse hashing for cross-modal similarity search. In: Proceedings of the 37th ACM International Conference on Research and Development in Information Retrieval, pp. 415–424 (2014)





# Continuous Patient-Centric Sequence Generation via Sequentially Coupled Adversarial Learning

Lu Wang, Wei Zhang<sup>(✉)</sup>, and Xiaofeng He<sup>(✉)</sup>

School of Computer Science and Software Engineering,  
East China Normal University, Shanghai, China  
joywanglulu@163.com, zhangwei.thu2011@gmail.com, xfhe@sei.ecnu.edu.cn

**Abstract.** Analyzing massive patient-centric Electronic Health Records (EHRs) becomes a key to success for improving health care and treatment. However, the amount of these data is limited and the access to EHRs is difficult due to the issue of patient privacy. Thus high quality synthetic EHRs data is necessary to alleviate these issues. In this paper, we propose a Sequentially Coupled Generative Adversarial Network (SC-GAN) to generate continuous patient-centric data, including patient state and medication dosage data. SC-GAN consists of two generators which coordinate the generation of patient state and medication dosage in a unified model, revealing the clinical fact that the generation of patient state and medication dosage data have noticeable mutual influence on each other. To verify the quality of the synthetic data, we conduct comprehensive experiments to employ these data on real medical tasks, showing that data generated from SC-GAN leads to better performance than the data from other generative models.

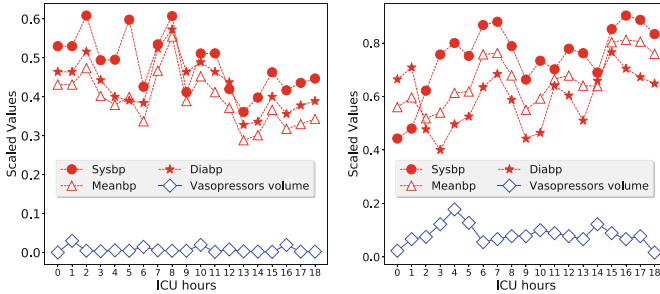
**Keywords:** Continuous data · Patient-centric sequence · Sequentially coupled adversarial learning

## 1 Introduction

The effective analysis of Electronic Health Records (EHRs) has the potential to improve clinical outcomes. However, since data of EHRs largely consists of personal medical information, it raises a significant privacy issue which discourages the public sharing of these data. In addition, the amount of these data is limited, because most of EHRs are self-governed by healthcare organizations which require formal collaborations and complex data usage agreements for even academic research purpose. Thus, the limited access to EHRs becomes the bottlenecks of advancing the field of healthcare [1] and hinders the development of medical data mining solutions.

Simulation is a standard practice for medical data generation and learning. Due to the complex hand-crafted rules of simulation design, automatically

generating synthetic data becomes a fashion for relieving privacy risks and the data scarcity issue [2, 3]. Specifically, deep generative models have recently been employed for releasing medical data mining [4, 5]. The generated medical data can be exploited for mitigating the risk of privacy and alleviate the data scarcity issue by data augmentation.



**Fig. 1.** The records in MIMIC-III show three blood pressure measurements of two sepsis patients and the vasopressors dosage prescribed to them, where vasopressors is used to counteract sepsis-induced vasodilation and elevate arterial pressure. The left patient who takes small dosage of vasopressors shows a declining blood pressure. The right patient who takes larger dosage has a rising blood pressure.

Generative Adversarial Networks (GANs) [6] train a generative model  $G$  and a discriminative model  $D$  simultaneously with antagonistic objectives which achieves promising results in generating realistic samples such as images [7–9], text [10, 11], etc. Only recently, a very few studies [4, 5, 12] apply GAN to synthesize medical data generation. However, [4, 5] focus on generating patient state data, ignoring medication dosage data which is another crucial type of patient-centric data. Although [12] enables the simultaneous generation of patient state and corresponding medications, it could only generate discrete values at a specified time, unable to generate continuous sequential medical data which is more in line with reality.

In this paper, we focus on the generation of continuous patient-centric sequence, mainly including patient state and corresponding medication dosage data, both of which play an important role in treatment recommendation [13, 14]. The key observation is that the generation of patient state and medication dosage data have significant mutual influence on each other. On one hand, doctors determine medication dosage mainly based on patients’ current state, leading to the generation of medication dosage influenced by the generation of patient state. On the other hand, the state of patients highly depend on the medication dosage they take. For example, various fluids and vasopressor dosage strategies have been proved to cause extreme variations in patient [15]. As shown in Fig. 1, the blood pressures of sepsis patients are affected by the dosage of vasopressor they take and the doctor also adjusts vasopressor dosage according to the blood pressures.

Inspired by the above observation, we propose a Sequentially Coupled GAN (SC-GAN) model to generate the state of patients and medication dosage together, which captures the interaction between them. Specifically, SC-GAN consists of coupled generators: one is leveraged to first generate current state of patients and the other further utilizes the acquired state to generate the corresponding medication dosage prescribed to each patient. As a result, the two generators are directly associated and trained jointly in a unified model to benefit each other.

Our main contributions are summarized as follows:

- We propose SC-GAN which consists of two interacted generators to produce both the state of patients and the corresponding medication dosage. The coupled generators capture the mutual influence of their generation, which is overlooked by previous studies. In addition, we adopt a hybrid loss trick which combines feature matching loss and standard generator loss to further improve the performance.
- Experiments on public available real-world EHRs show the treatment recommendation model trained on the synthetic data generated by our model achieves better performance than state-of-arts and incorporating the synthetic data into the real datasets can further improve the performance.

## 2 Related Work

In this section, we overview the related studies from two aspects: sequentially generative adversarial networks and medical data generation.

### 2.1 Sequentially Generative Adversarial Networks

Generative adversarial networks are generative models with the mechanism of adversarial training, where the goal of  $D$  is to discriminate between real data and the samples generated by  $G$ , and the goal of  $G$  is to fool  $D$  with generated realistic data. Although GANs have achieved impressive success in image generation [16], there are limited studies using GANs to produce sequential data. The most conventional methods with this regard are recurrent neural networks (RNNs). RNNs have been utilized to generate sequential discrete tokens (e.g., machine translation [17]) and continuous values (e.g., music data [18,19]). The most common objective for optimizing RNNs is based on maximum likelihood. However, utilizing this criterion to generate sequence data has been argued to suffer from the *exposure bias* [20]. In contrast, GANs work well to mitigate this problem. SeqGAN [10] extends GAN with RNN to generate sequences of discrete word tokens via policy gradient. C-RNN-GAN [21] trains RNNs with adversarial training for continuous music generation, which is a pioneering study to generate sequential and continuous data. Several methods also employ convolutional neural networks (CNNs) for generating audios and images. Although using convolutional GANs to generate sequential data may have faster training speed than recurrent GANs, it loses the Markov property of trajectory samples.

## 2.2 Medical Data Generation

The generated medical data helps to build predictive systems in the medical domain, such as predicting the patient-specific trajectories (e.g., Albumin, Arterial pH, Calcium, etc.) or recommending treatments for a given patient. Although the most commonly acceptable approach to generate EHRs dataset for sharing is de-identification [22], the individual information of the patients can be re-identified through residual distinguishable patterns [12]. For example, re-identifying lab tests, demographics, and genomic variants. Generating synthetic data becomes an alternative approach to reduce the privacy risk.

Followed by the successful applications of GANs mentioned above, a set of studies begin to employ GANs to generate medical data for sharing. Li et al. [23] proposed a hybrid GAN to generate text reports for medical image with high-level and low-level modules. Most related to this work, Yahi et al. [5] utilized RNNs with adversarial learning to generate the laboratory test time series data. Nevertheless, it overlooks the fact that the patients' state are highly influenced by the medications they take. On the other hand, Beaulieu et al. [4] employed the Auxiliary Classifier Generative Adversarial Network (AC-GAN) [9] to generate real-valued state of patient to provide a freely accessible public version for discovery-oriented analysis. Esteban et al. [24] used a recurrent GAN to generate real-valued time-series state of patients, which also only considered the generation of state as [9]. Edward et al. [12] combined autoencoder with GAN to generate the discrete variables such as diagnosis, medication and procedure codes. However, it only considers one-step generation instead of sequential generation.

To reflect the clinical fact, we propose SC-GAN to capture the interactions between continuous state of patients and the medication dosage they take. Specifically, SC-GAN designs coupled generators to produce the interdependent state and medication dosage. Note that our proposed model is significantly different from the Multi-Generator generative adversarial net [25] which utilizes multiple generators to generate one single type of data.

## 3 Preliminaries

In this section, we first briefly introduce the data of continuous patient-centric sequence and some basic notations, followed by the problem definition and the description of GANs with its main variants.

### 3.1 Data Description and Notations

The major goal of this paper is to generate patient-centric medical sequence data. Unlike previous studies [4, 5, 12], we focus on generating continuous values with consideration of their mutual interactions. Generally, the numerical data could be categorized into two aspects: (1) patient state data which includes lab tests, vital signs, intake, etc; (2) medication dosage data.

Since both patient state data and medication dosage data present sequential characteristics, we assume  $\mathbf{s}_t$  denotes the state data of a patient at time step  $t$ . Noting that to simplify the notations, we omit the subscript about a specific patient and can apply these notations to different patients. Correspondingly, we represent the medication dosage data as  $\mathbf{a}_t$ . For example, in the clinical practice, doctors always design proper medication dosage for patients based on their current state [13,26]. Meanwhile, patient state would vary after taking the medication dosage.

Based on this intuition, we implement SC-GAN by ensuring that the current state data  $\mathbf{s}_t$  is generated based on the previous state data  $\mathbf{s}_{t-1}$  and the medication dosage data  $\mathbf{a}_{t-1}$ , while in turn the medication dosage data  $\mathbf{a}_t$  is produced based on the input of the current state  $\mathbf{s}_t$ .

### 3.2 Problem Definition

The detail of the problem studied in this paper is described in Problem 1.

*Problem 1 [Continuous Patient-centric Sequence Generation].* For a patient with given disease and medication, we aim at generating the sequential state  $\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_T\}$  and corresponding dosage  $\mathbf{A} = \{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_T\}$  with the consideration of their mutual interactions.

### 3.3 Basics of GANs

Generative Adversarial Networks are generative models which consist of two neural networks: Discriminator Net  $D(\mathbf{x}; \theta_d)$  and Generator Net  $G(\mathbf{z}; \theta_g)$ , where  $\mathbf{z}$  is a random noise.  $D(\mathbf{x})$  indicates the probability that  $\mathbf{x}$  comes from a real data distribution. To discriminate the real data from synthetic data, it maximizes the probability of real data and minimizes the probability of synthetic data generated from  $G$ . In contrast,  $G$  has the opposite goal which is to generate realistic data to make  $D$  indistinguishable. That is,  $D$  and  $G$  play the following minimax game to reach a Nash equilibrium:

$$\begin{aligned} \min_G \max_D V(D, G) = & \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})} [\log D(\mathbf{x})] \\ & + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))] \end{aligned} \quad (1)$$

where  $p_{data}$  represents the distribution of real data.  $p_z(\mathbf{z})$  denotes the distribution of noises where normal distribution  $\mathcal{N}(0, 1)$  and uniform distribution  $\mathcal{U}(0, 1)$  are the common choices.  $D$  and  $G$  are iteratively optimized.

Unfortunately, if the discriminator  $D$  is excessively strong, then Eq. 1 may be unable to give sufficient gradient for  $G$  to update its parameters. Instead of using the objective which maximizes the predicted probability of the discriminator, Salimans et al. proposed feature matching [27] which is a new strategy to optimize  $G$ .

Feature matching aims at generating samples actually fall into the real data manifold. It also encourages greater variance in  $G$  as well as prevents it from

overtraining on the current discriminator. Formally, the feature matching loss is described as follows:

$$L_G = \|\mathbb{E}_{x \sim p_{data}} \mathbf{f}(x) - \mathbb{E}_{z \sim p_z(z)} \mathbf{f}(G(z))\|_2^2 \quad (2)$$

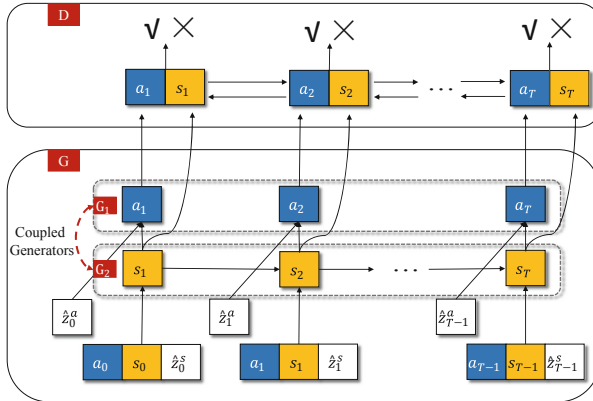
where  $\mathbf{f}(x)$  is the representation of the last layer before the final classification of  $D$ .

Inspired by the above methodologies, we provide our SC-GAN model by coupling two sequential generators in a unified generative adversarial network, capturing the mutual interaction in continuous patient-centric medical sequence.

## 4 Methodology

### 4.1 Overview of SC-GAN

SC-GAN aims at generating synthetic patient-centric medical data which consists of trajectory state data  $\mathbf{S}$  ( $\mathbf{S} = \{s_1, s_2, \dots, s_T\}$ ) of patients and corresponding medication dosage  $\mathbf{A}$  ( $\mathbf{A} = \{a_1, a_2, \dots, a_T\}$ ) during treatment process. Specifically, the medication dosage data  $a_t$  at time step  $t$  is generated based on the current state data  $s_t$  of a patient, and the current state  $s_t$  is generated based on the previous state data  $s_{t-1}$  and previous medication dosage data  $a_{t-1}$ . As shown in Fig. 2, we establish two generators  $G_1$  and  $G_2$  for generating  $\mathbf{S}$  and  $\mathbf{A}$  respectively, where  $a_t$  is with the input of random noise  $\hat{z}_t^a$ , state  $s_t$ , and  $s_t$  is with the input of random noise  $\hat{z}_{t-1}^s$ ,  $a_{t-1}$  and  $s_{t-1}$ . In other words, the two generators interact with each other to generate these two categories of data. To implement the discriminator, we set a classification task to distinguish the real and synthetic data at each time step.



**Fig. 2.** The general framework of Sequentially Coupled GAN. This model consists of three main components: a discriminator (a 2-layer bidirectional LSTM) and two interdependent generators (two 2-layer LSTMs).  $s_t$  indicates trajectory state of the patients at time-point  $t$ .  $a_t$  represents the medication dosage used for patients.  $\hat{z}_t^a$  and  $\hat{z}_t^s$  are random noises of the medication dosage and patient state respectively. The correct symbol ( $\checkmark$ ) indicates the discriminator determines the data is real while the incorrect symbol ( $\times$ ) indicates the data is judged as synthetic.

## 4.2 Coupled Generators

The goal of the coupled generators  $G_1$  and  $G_2$  is to generate realistic synthetic medical records of patients to maximize the probability of  $D$  for letting  $D$  make a misjudgment. Both  $G_1$  and  $G_2$  have two layers of long short-term memory networks (LSTM) [28]. The reasons are that: (1) LSTM is capable of exhibiting temporal dynamics compared to feed-forward networks and CNNs; (2) LSTM utilizes three gates to protect and control the cell state, which mitigates the gradient vanishing and exploding problems compared to RNNs.

$G_1$  generates medication dosage data  $(\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_T)$  with the input of sequential continuous state data  $(\mathbf{s}_0, \mathbf{s}_1, \dots, \mathbf{s}_{T-1})$  and a random noise sequence  $(\hat{\mathbf{z}}_0^a, \hat{\mathbf{z}}_1^a, \dots, \hat{\mathbf{z}}_{T-1}^a)$ . Formally, at each time step  $t$ , the input  $\mathbf{z}_t^a$  of  $G_1$  is the concatenation of  $\mathbf{s}_t$  and  $\hat{\mathbf{z}}_t^a$ :

$$\mathbf{z}_t^a \leftarrow [\mathbf{s}_t; \hat{\mathbf{z}}_t^a], \quad (3)$$

$$\mathbf{a}_t = G_1(\mathbf{z}_t^a), \quad (4)$$

where  $\mathbf{s}_t$  is the output of  $G_2$  at time step  $t$  and  $\hat{\mathbf{z}}_t^a \in \mathcal{U}(0, 1)$ .

$G_2$  is leveraged to generate the patient state data  $\mathbf{s}_t$  with the input of previous state  $\mathbf{s}_{t-1}$ , the medication dosage data  $\mathbf{a}_{t-1}$ , and the current random noise  $\hat{\mathbf{z}}_t^s$ . In other words, at each time step of  $G_2$ , the input  $\mathbf{z}_t^s$  at time step  $t$  is the concatenation of  $\mathbf{s}_{t-1}$ ,  $\mathbf{a}_{t-1}$ , and  $\hat{\mathbf{z}}_t^s$ :

$$\mathbf{z}_t^s \leftarrow [\mathbf{s}_{t-1}; \mathbf{a}_{t-1}; \hat{\mathbf{z}}_t^s], \quad (5)$$

$$\mathbf{s}_{t-1} = G_2(\mathbf{z}_{t-1}^s), \mathbf{a}_{t-1} = G_1(\mathbf{z}_{t-1}^a), \quad (6)$$

where  $\mathbf{s}_{t-1}$  is the output of  $G_2$  at time step  $t-1$ ,  $\mathbf{a}_{t-1}$  is the output of  $G_1$  at time step  $t-1$ , and  $\hat{\mathbf{z}}_t^s$  is also a uniform random value in  $[0, 1]$ .

As shown in Eqs. 4 and 6, the outputs of  $G_1$  and  $G_2$  are also the inputs of  $G_2$  and  $G_1$ . Combining these two generators together and differentiating different patients, we minimize the following objective function to train these generators:

$$L_G = \frac{1}{N} \frac{1}{T} \sum_{i=1}^N \sum_{t=1}^T \log(1 - D(G(\mathbf{z}_{i,t}))) \quad (7)$$

$$G(\mathbf{z}_{i,t}) = [G_1(\mathbf{z}_{i,t}^a); G_2(\mathbf{z}_{i,t}^s)] \quad (8)$$

where  $N$  is the number of patients,  $T$  is the time length of the patient record. That is, the optimal generator  $G$  should generate both realistic patient state data and medication dosage data at the same time.

As for training, we first conduct a supervised pretraining step for SC-GAN. To avoid training an excessively strong discriminator to hamper the generators optimization, we only pretrain the coupled generators by utilizing the least square loss to generate the sample of the next time step. The objective function for the pretraining step is defined as follows:

$$L_{pretrain} = \frac{1}{N} \frac{1}{T} \sum_{i=1}^N \sum_{t=1}^T (\|\mathbf{s}_{i,t}^{real} - G_1(\mathbf{z}_{i,t}^a)\|_2^2 + \|\mathbf{a}_{i,t}^{real} - G_2(\mathbf{z}_{i,t}^s)\|_2^2) \quad (9)$$

where  $\mathbf{s}_{i,t}^{real}$  and  $\mathbf{a}_{i,t}^{real}$  indicate the real state and dosage data of patient  $i$  at time step  $t$ , respectively. In the pretraining step, the coupled generators are with the input of concatenation of the random noise and the real data from training set.

During adversarial training, the goal of generator  $G$  is to produce samples which can cheat  $D$ . Thus  $G$  could map  $p_z(\mathbf{z})$  to only a few and low-volume regions. That is,  $G$  may produce the same synthetic data. This problem is called *mode collapse*. To improve the variance of  $G$  and address the instability of GANs, we combine the feature matching loss shown in Eq. 2 with the standard generator loss through a weighted linear fusion, which is defined as follows:

$$L_G = \frac{1}{N} \frac{1}{T} \sum_{i=1}^N \sum_{t=1}^T \left( \lambda_{fm} (\|\mathbf{f}(\mathbf{s}_{i,t}^{true}) - \mathbf{f}(G_1(\mathbf{z}_{i,t}^a))\|_2^2 + \|\mathbf{f}(\mathbf{a}_{i,t}^{true}) - \mathbf{f}(G_2(\mathbf{z}_{i,t}^s))\|_2^2) + \lambda_{adv} (\log(1 - D(G(\mathbf{z}_{i,t}))) ) \right) \quad (10)$$

where  $\lambda_{fm} \in [0, 1]$  and  $\lambda_{adv} \in [0, 1]$  are the weights of feature matching loss and standard generator loss, respectively, and they are tuned empirically based on the test performance.  $\mathbf{f}$  is the representation of the last layer before final classification of  $D$ .

### 4.3 Discriminator

The goal of discriminator  $D(\mathbf{x})$  is to correctly judge the real data and the generated synthetic data. SC-GAN classifies the data into real or synthetic at each time step, to simplify the procedure of directly discriminating the whole sequence of data. Specifically,  $D$  is trained to minimize the negative cross-entropy loss between the real sequential patient-centric records ( $[\mathbf{s}_1; \mathbf{a}_1], [\mathbf{s}_2; \mathbf{a}_2], \dots, [\mathbf{s}_T; \mathbf{a}_T]$ ) and the generated data ( $G(\mathbf{z}_1), G(\mathbf{z}_2), \dots, G(\mathbf{z}_T)$ ).  $D$  has a 2-layer bidirectional LSTM, which could integrate the context in both directions. Finally, the loss of the discriminator can be described as follows:

$$L_D = -\frac{1}{N} \frac{1}{T} \sum_{i=1}^N \sum_{t=1}^T \left( \log D(\mathbf{x}_{i,t}) + \log(1 - D(G(\mathbf{z}_{i,t}))) \right) \quad (11)$$

$$\mathbf{x}_{i,t} = [\mathbf{s}_t; \mathbf{a}_t], G(\mathbf{z}_{i,t}) = [G_1(\mathbf{z}_{i,t}^a); G_2(\mathbf{z}_{i,t}^s)] \quad (12)$$

where  $\mathbf{x}_{i,t}$  consists of the real state of patient  $i$  and his/her medication dosage data at time step  $t$ .

## 5 Experiments

In this section, we conduct extensive experiments on two distinct patient-centric datasets extracted from real-world EHRs, aiming at demonstrating the data generated by SC-GAN is better than those of several comparative models. The source code will be released with the publication of this paper for relevant study.



## 5.1 Dataset Description and Preprocessing

**Patient Cohort.** The experiments are conducted on a real-world EHRs, namely the Multiparameter Intelligent Monitoring in Intensive Care (MIMIC-III v1.4) database [29]. MIMIC-III encompasses a population of 43K patients and 474 million patient-centric state observations in intensive care units (ICUs) during 2001 and 2012. Based on MIMIC-III, we construct two distinct disease datasets: sepsis and diabetes. Sepsis is a main cause of mortality in ICUs [13] and a great deal of studies try to find optimal treatment dosage for sepsis. Diabetes is a lifestyle-related chronic disease and glycemic control with proper dosage is essential for diabetes [30].

We extract sepsis patients conforming to the Sepsis-3 criteria [31] and extract diabetes, mycosis and isoniazid patients with ICD-9 codes for diabetes. We summarize the basic statistics of the extracted patients in Table 1. We randomly divide the dataset for training, validation, and testing sets by the proportion of 80:10:10.

**Table 1.** Basic statistics of the two datasets

Description	Sepsis-3	Diabetes	Mycosis	Isoniazid
% Female	43.6	32.1	44.8	41.9
Mean age	66.6	76.8	62.7	68.2
Hours in ICU	59.3	82.7	63.4	79.5
Total population	13,773	5,538	6,722	3,245

**State of Patients.** For each patient, we extract relevant physiological parameters as his/her state, such as laboratory tests, vital signs and output events. The details of these features are shown in Table 2. We aggregate the data into windows of 4 h to obtain patient-centric multidimensional time series data. The missing variables are imputed by k-nearest neighbors and the records with more than 10 missing variables are removed. We rescale each feature at each time step independently to the range [0, 1].

**Medication Dosage of Patients.** We select intravenous fluids (IV fluids) and vasopressor as the main medications of sepsis patients and choose insulin for diabetes patient. Because the dosage of these medications have been verified to highly affect the state of diabetes patients and sepsis patients, and even their mortality in ICU.

**Table 2.** Description of the trajectory state of patients.

Laboratory tests	Albumin, Arterial_pH, Calcium, Glucose, Partial Thromboplastin Time, Potassium, SGPT, Arterial Blood Gas, Blood Urea Nitrogen, Chloride, International Normalized Ratio, Sodium, Ionised Calcium, Arterial Lactate, CO2, Creatinine, Prothrombin Time, SGOT Platelets Count, Total bilirubin, White Blood Cell Count, Magnesium
Vital signs	Diastolic Blood Pressure, PaCO2, Systolic Blood Pressure, PaO2, Mean Blood Pressure, FiO2, PaOFiO2 ratio, Respiratory Rate, Temperature (Celsius), SaO2, Heart Rate, SpO2, Arterial_BE
Output event	Total Fluid Output

## 5.2 Models for Comparison

- **SeqGAN** [10]: SeqGAN considers the sequence generation procedure as a sequential decision making process, where the generator represents a reinforcement learning agent and the discriminator indicates an evaluator to guide the generator. We replace the last layer of the generator to produce continuous medical data.
- **C-RNN-GAN** [21]: A method employs GANs for generating sequential continuous music. It utilizes an LSTM to represent the generator and a bidirectional LSTM as the discriminator. The discriminator performs classification for each sequence.
- **RCGAN** [5]: It has a similar architecture as C-RNN-GAN, except that the outputs of the generator are not fed back into the inputs and the discriminator conducts a discrimination at each time step of the sequence.
- **Imitation (RNN)**: A RNN based model which has the same structure as the generators of SC-GAN. It is also used as the pre-training process of SC-GAN.
- **SC-GAN**: Proposed model, which generates the state and medication dosage of the patients simultaneously. To reflect the clinical facts, SC-GAN uses coupled generators to produce state and medication dosage respectively, where two generators are interacted with each other.
- **SC-GAN (one G)**: A variant of SC-GAN, the state of patients and the corresponding medication dosage are produced using one single generator without interaction.
- **SC-GAN** ( $\lambda_{fm} = 0$ ): A variant of SC-GAN, we set  $\lambda_{fm} = 0$  and  $\lambda_{adv} = 1$  in Eq. 10.
- **SC-GAN** ( $\lambda_{adv} = 0$ ): A variant of SC-GAN, we set  $\lambda_{fm} = 1$  and  $\lambda_{adv} = 0$ .

Although SeqGAN and C-RNN-GAN are not designed to generate medical data, they are employed for generating sequential data which resembles our data type and thus can be easily adapted to the medical domain.

### 5.3 Quantitative Evaluation for Synthetic Data

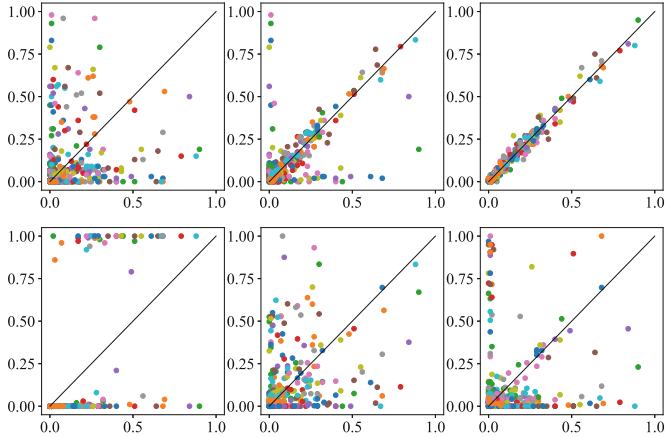
It is challenging to evaluate the performance of GANs. Human judgment can be a candidate choice, but it is impractical and costly especially for medical records. We conduct both quantitative and qualitative analysis to evaluate the generated data, such as dimension-wise probability [12], treatment recommendation task evaluation, and Pearson correlations of real data and synthetic data [4], etc.

**Dimension-Wise Probability.** This metric is used to measure how the distribution of generated patient data matches the real data distribution. To be specific, we discretize the values of the generated/real patient state features and dosage with the window size of 0.1. Thus each feature or dosage has eleven value slots (i.e., 0, 0.1, ..., 1). Then we calculate the probabilities of the occurrences for each features in different slots. Take the  $i$ -th feature and  $m$ -th slot as an example. Suppose the corresponding probability of real training data denoted as  $p_{i,m}$  and the value from synthetic or real test data denoted as  $\hat{p}_{i,m}$ . We regard  $(p_{i,m}, \hat{p}_{i,m})$  as a point with x-coordinate  $x = p_{i,m}$  and y-coordinate  $y = \hat{p}_{i,m}$ . We collect all points regarding different datasets and plot them in Fig. 3. This probability reflects the distribution of each value of the features and is a very important statistical indicator. Intuitively, if more points appear near the line  $x = y$ , it means the feature value distribution of  $\hat{p}_{i,m}$  better matches the feature value distribution of  $p_{i,m}$ .

From the results, we can see SeqGAN has poor performance, where the distribution of the generated values is significantly different from that of the real data. C-RNN-GAN and RCGAN generate a set of values with high/low probability while the corresponding probability of the true data is reverse. The dimension-wise probability performance increases as we consider the dependence between medication dosage and state (compared to C-RNN-GAN and RCGAN). SC-GAN also shows better performance than Imitation, which demonstrates that the adversarial learning mechanism (SC-GAN) could work better than maximum likelihood mechanism (Imitation) in generating sequential clinical data.

**Treatment Recommendation Task Evaluation.** It is an important issue to utilize large amount of medical records (e.g., EHRs) to improve the quality of medical treatment especially to design proper dosage for patients [13, 26]. In this section, we conduct a treatment recommendation model which designs proper medication dosage for patients to evaluate whether the synthetic data could be used for real applications, inspired by [32] which has three layers with size 58, 58 and 5 and the activation function Relu. The input is patient state and the output is the medication dosage class recommended for patients. It is trained on synthetic data and tested on real data. Two distinct generated patient-centric data: sepsis and diabetes datasets are utilized for the task. Following the previous work, we discretized the dosage of medications into 5 medication space. Thus, the number of treatment/dosage classes is five.

Table 3 shows the precision and AUROC of the recommendation results on different synthetic data. *True data* indicates that the recommendation model is



**Fig. 3.** Dimension-wise probability results. The top three figures, from left to right, represent the performance of Imitation, SC-GAN and True data while the below are SeqGAN, RCGAN and C-RNN-GAN. Different color corresponds to a feature of the patient. The x-axis indicates the probability for the real training data, and y-axis represents the probability for the synthetic test data or the real test data. The diagonal line is the optimal performance where the training data and test data exactly match. (Color figure online)

trained with true data and test on the true data. We randomly select 11,018 sepsis samples and 4,430 diabetes samples from generated data with same amount as real data in the training step. SeqGAN shows poor performance because it is designed for generating discrete data which may not be suitable for continuous data. RCGAN also performs not very well due to the reason that the outputs of its generator are not utilized as input for modeling. Imitation (RNN) and C-rnn-gan perform better than RCGAN, showing the benefits of leveraging outputs from last time step as current input and using RNN for modeling sequence data. SC-GAN outperforms the other baselines. The reason is that it considers the mutual interactions between drug dosage and state of patients, which reflects the clinical practice (compared with C-rnn-gan, RCGAN and SC-GAN (one)). It also utilizes the adversarial training mechanism to generate more realistic data and mitigate the exposure bias of maximum likelihood methods (compared with Imitation). By integrating both the standard loss and feature matching loss, SC-GAN achieves better performance (compared with SC-GAN ( $\lambda_{adv} = 0$ ) and SC-GAN ( $\lambda_{fm} = 0$ )).

**Data Augmentation Test.** In addition to only using synthetic data, we train the treatment recommendation model on augmented dataset, where the synthetic data and true data are combined with the ratio of 2 : 3. This experiment is utilized to show whether the generated data can bring more information and help alleviate the clinical data scarcity issue.

**Table 3.** Precision and AUROC of treatment recommendation task while trained on real data or synthetic data and testing on real data (%).

Methods	Sepsis-3				Diabetes		Mycosis		Tuberculosis	
	IV fluids		Vasopressor		Insulin		Fluconazole		Isoniazid	
	Pre.	AUROC	Pre.	AUROC	Pre.	AUROC	Pre.	AUROC	Pre.	AUROC
True data	<b>36.2</b>	<b>60.1</b>	<b>34.6</b>	<b>58.7</b>	<b>62.0</b>	<b>76.2</b>	<b>36.2</b>	<b>60.1</b>	<b>34.6</b>	<b>58.7</b>
C-rnn-gan	26.8	53.3	24.1	52.6	53.4	71.3	24.4	51.8	23.1	51.2
SeqGAN	18.7	50.3	18.4	50.1	41.5	66.8	17.7	50.5	17.3	50.1
Imitation (RNN)	27.1	53.7	25.3	52.8	54.1	71.8	25.3	52.5	24.0	51.4
RCGAN	25.6	52.6	23.5	51.3	51.9	70.3	24.1	51.6	22.8	50.9
SC-GAN	<b>31.4</b>	<b>57.1</b>	<b>29.3</b>	<b>55.9</b>	<b>56.3</b>	<b>73.1</b>	<b>30.3</b>	<b>56.2</b>	<b>27.3</b>	<b>53.6</b>
SC-GAN (one G)	29.9	56.2	27.9	54.1	52.9	72.2	28.2	55.1	24.7	52.3
SC-GAN ( $\lambda_{adv} = 0$ )	29.3	56.1	27.3	53.6	52.8	72.0	28.0	54.9	24.3	52.1
SC-GAN ( $\lambda_{fm} = 0$ )	28.8	55.9	27.0	53.1	52.5	71.8	27.8	54.7	24.2	52.1

As shown in Table 4, the data generated by SeqGAN and RCGAN seems to make no contribution to the real data because results show no improvements over those of only employing real data, showing no additional useful information is generated. Imitation and C-rnn-gan behave better for their contribution to improving recommendation performance. Finally, by adding the data generated by SC-GAN, the recommendation model achieves significantly better results, demonstrating the generated data indeed provide additional useful information to complement true data and it is good for data augmentation.

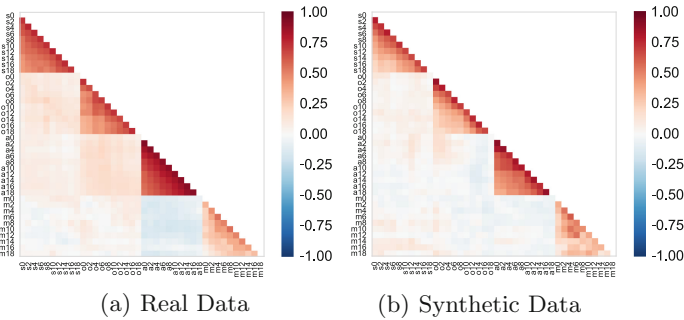
**Table 4.** Precision and AUROC of medication dosage recommendation task trained with both synthetic and real training data and tested on real data (%).

Methods	Sepsis-3				Diabetes		Mycosis		Tuberculosis	
	IV fluids		Vasopressor		Insulin		Fluconazole		Isoniazid	
	Pre.	AUROC	Pre.	AUROC	Pre.	AUROC	Pre.	AUROC	Pre.	AUROC
True data	36.2	60.1	34.6	58.7	62.0	76.2	48.5	69.2	47.3	68.7
C-rnn-gan	36.9	60.4	33.4	58.1	62.9	76.6	36.6	63.7	32.4	61.3
SeqGAN	32.2	58.3	30.0	57.2	54.3	73.2	25.3	57.7	24.8	52.0
Imitation (RNN)	37.3	60.7	34.8	<b>59.2</b>	63.2	76.9	36.5	64.8	33.6	61.6
RCGAN	35.4	59.7	33.2	58.0	61.4	76.3	36.1	63.4	32.3	61.1
SC-GAN	<b>38.6</b>	<b>61.4</b>	<b>35.7</b>	<b>59.2</b>	<b>64.6</b>	<b>77.3</b>	<b>39.2</b>	<b>65.1</b>	<b>34.4</b>	<b>62.5</b>
SC-GAN (one G)	37.1	60.7	33.8	59.3	62.7	76.4	37.3	62.5	32.1	61.3
SC-GAN ( $\lambda_{adv} = 0$ )	36.5	59.8	33.1	58.6	61.8	76.0	37.1	62.3	31.8	61.2
SC-GAN ( $\lambda_{fm} = 0$ )	36.2	59.3	32.9	58.0	61.4	75.7	36.8	62.1	31.5	61.0

## 5.4 Qualitative Evaluation for Synthetic Data

**Pairwise Pearson Correlation.** Following [4], we adopt Pearson correlation coefficient (PCC) [33] to obtain Pearson correlation structures of feature pairs for real data and synthetic data, respectively. The value of PCC ranges from  $+1$  to  $-1$ , where  $1$  represents complete positive correlation, and  $-1$  corresponds to complete negative correlation. We select three features of the patients (systolic blood pressure, spo2 and Arterial\_BE) and the dosage of Intravenous Fluids to conduct Pearson correlation experiments. The Pearson correlation structures of the real data is in Fig. 4(a) and generated data is in Fig. 4(b).

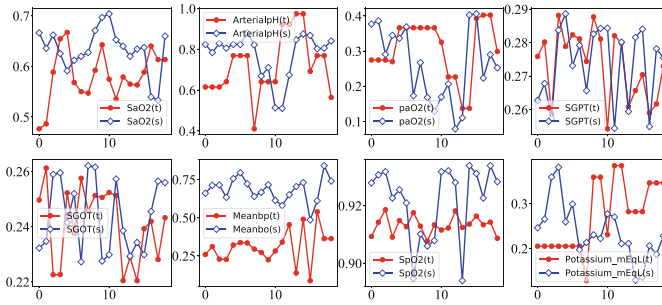
As shown in Fig. 4,  $s$ ,  $o$ ,  $a$ ,  $m$  represent systolic blood pressure, spO<sub>2</sub>, Arterial\_BE and Intravenous Fluids respectively. The numbers 0–18 indicate the ICU stay length (hour) of the sepsis patients. Here we only extract ten time steps for comparison due to the space limitation. Both the synthetic data and real data show the Intravenous Fluids has positive correlation with systolic blood pressure and spo2. But for Arterial\_BE and Intravenous Fluids, the synthetic data shows weaker correlation than the real data. The synthetic data generated by SC-GAN shows a little different result compared to the true data. The main trends of the results remain consistent.



**Fig. 4.** Pairwise Pearson correlation (PPC) between time series features.

**Generated Patient-Centric Data.** To conduct a qualitative evaluation for synthetic data, we randomly select eight features generated by SC-GAN to intuitively compare the difference between them and the true data. We also invite two clinicians to evaluate the results.

For most of the generated data, the clinicians could not judge they are synthetic, except for paO<sub>2</sub>. This is because paO<sub>2</sub> involves frequent variation. As shown in Fig. 5, the trends of the generated features are not regular. Because the state of these patients can change significantly as the time goes on. However, the mean of these data could be concentrated. Figure 5 shows that the values of different patient features vary in different regions. The synthetic data produced by SC-GAN obtains the similar values regions as the true data.



**Fig. 5.** Generated trajectory data lasting from initial visit to  $19 * 4$  h for specified features (“t” indicates the true data and “s” indicates the synthetic data)

## 6 Conclusion

In this paper, we have proposed SC-GAN to generate sequential and continuous medical data including the state of patients and the dosage of medications the patients take. The main novelty of the model is the coupled generators which coordinate the generation of patient state and medication dosage to capture the mutual interactions between medications and patient state. The comprehensive experiments on the real world EHRs dataset demonstrate the data generated by SC-GAN can not only gain performance close to the real data on treatment recommendation task, but also be useful for data augmentation.

**Acknowledgements.** This work is partially supported by the National Key Research and Development Program of China under Grant No. 2016YFB1000904, NSFC (61702190, U1609220).

## References

1. Gostin, L.O., et al.: Beyond the HIPAA Privacy Rule: Enhancing Privacy. Improving Health Through Research. National Academies Press, Washington, DC (2009)
2. McLachlan, S., Dube, K., Gallagher, T.: Using the caremap with health incidents statistics for generating the realistic synthetic electronic healthcare record. In: Healthcare Informatics (ICHI), pp. 439–448 (2016)
3. Buczak, A.L., Babin, S., Moniz, L.: Data-driven approach for creating synthetic electronic medical records. *BMC Med. Inform. Decis. Making* **10**, 59 (2010)
4. Beaulieu-Jones, B.K., et al.: Privacy-preserving generative deep neural networks support clinical data sharing, p. 159756. *BioRxiv*, C.S. (2017)
5. Yahi, A., Vanguri, R., Elhadad, N., Tatonetti, N.P.: Generative adversarial networks for electronic health records: a framework for exploring and evaluating methods for predicting drug-induced laboratory test trajectories. In: NIPS (2017)
6. Goodfellow, I., et al.: Generative adversarial nets. In: NIPS, pp. 2672–2680 (2014)
7. Isola, P., Zhu, J., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: CVPR, pp. 5967–5976 (2017)

8. Gregor, K., Danihelka, I., Graves, A., Rezende, D.J., Wierstra, D.: DRAW: a recurrent neural network for image generation. In: ICML, pp. 1462–1471 (2015)
9. Odena, A., Olah, C., Shlens, J.: Conditional image synthesis with auxiliary classifier GANs. In: ICML, pp. 2642–2651 (2017)
10. Yu, L., Zhang, W., Wang, J., Yu, Y.: SeqGAN: sequence generative adversarial nets with policy gradient. In: AAAI, pp. 2852–2858 (2017)
11. William, F., Goodfellow, I., Dai, A.M.: MaskGAN: better text generation via filling in the .. In: ICLR (2018)
12. Choi, E., Biswal, S., Malin, B., Duke, J., Stewart, W.F., Sun, J.: Generating multi-label discrete electronic health records using generative adversarial networks. *Machine Learning for Healthcare* (2017)
13. Raghu, A., Komorowski, M., Celi, L.A., Szolovits, P., Ghassemi, M.: Continuous state-space models for optimal sepsis treatment: a deep reinforcement learning approach. In: *Proceedings of the Machine Learning for Health Care*, pp. 147–163 (2017)
14. Wang, L., Zhang, W., He, X., Zha, H.: Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. In: KDD, pp. 2447–2456. ACM (2018)
15. Waechter, J., et al.: Interaction between fluids and vasoactive agents on mortality in septic shock: a multicenter, observational study. *Criti. Care Med.* **42**(10), 2158–2168 (2014)
16. Denton, E.L., et al.: Deep generative image models using a Laplacian pyramid of adversarial networks. In: NIPS, pp. 1486–1494 (2015)
17. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: NIPS, pp. 3104–3112 (2014)
18. Casella, P., Paiva, A.: MAgentA: an architecture for real time automatic composition of background music. In: de Antonio, A., Aylett, R., Ballin, D. (eds.) IVA 2001. LNCS (LNAI), vol. 2190, pp. 224–232. Springer, Heidelberg (2001). [https://doi.org/10.1007/3-540-44812-8\\_18](https://doi.org/10.1007/3-540-44812-8_18)
19. Zhu, H., et al.: Xiaoice band: a melody and arrangement generation framework for pop music. In: KDD, pp. 2837–2846 (2018)
20. Bengio, S., Vinyals, O., Jaitly, N., Shazeer, N.: Scheduled sampling for sequence prediction with recurrent neural networks. In: NIPS, pp. 1171–1179 (2015)
21. Mogren, O.: C-RNN-GAN: continuous recurrent neural networks with adversarial training. CoRR abs/1611.09904 (2016)
22. Office for Civil Rights: Guidance regarding methods for de-identification of protected health information in accordance with the health insurance portability and accountability act (HIPAA) privacy rule. U.S. Department of Health and Human Services (2013)
23. Li, C.Y., Liang, X., Hu, Z., Xing, E.P.: Hybrid retrieval-generation reinforced agent for medical image report generation. arXiv preprint [arXiv:1805.08298](https://arxiv.org/abs/1805.08298) (2018)
24. Esteban, C., Hyland, S.L., Rätsch, G.: Real-valued (medical) time series generation with recurrent conditional GANs. arXiv preprint [arXiv:1706.02633](https://arxiv.org/abs/1706.02633) (2017)
25. Hoang, Q., Nguyen, T.D., Le, T., Phung, D.: Multi-generator generative adversarial nets. In: ICLR (2017)
26. Wang, L., Zhang, W., He, X., Zha, H.: Personalized prescription for comorbidity. In: Pei, J., Manolopoulos, Y., Sadiq, S., Li, J. (eds.) DASFAA 2018. LNCS, vol. 10828, pp. 3–19. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-91458-9\\_1](https://doi.org/10.1007/978-3-319-91458-9_1)
27. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training GANs. In: NIPS, pp. 2234–2242 (2016)



28. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.* **9**(8), 1735–1780 (1997)
29. Johnson, A.E., et al.: MIMIC-III, a freely accessible critical care database. *Sci. Data* **3**, 160035 (2016)
30. Weng, W.H., Gao, M., He, Z., Yan, S., Szolovits, P.: Representation and reinforcement learning for personalized glycemic control in septic patients. In: *NIPS Workshop* (2017)
31. Singer, M., et al.: The third international consensus definitions for sepsis and septic shock (sepsis-3). *JAMA* **315**(8), 801–810 (2016)
32. Bajor, J.M., ALasko, T.: Predicting medications from diagnostic codes with recurrent neural networks. In: *ICLR* (2017)
33. Pearson, K.: Notes on regression and inheritance in the case of two parents. *Proc. R. Soc. London* **58**, 240–242 (1895)



# DMMAM: Deep Multi-source Multi-task Attention Model for Intensive Care Unit Diagnosis

Zhenkun Shi<sup>1,2,3</sup> , Wanli Zuo<sup>1,2</sup>, Weitong Chen<sup>3</sup>, Lin Yue<sup>1,4</sup>, Yuwei Hao<sup>1,2</sup> ,  
and Shining Liang<sup>1,2</sup> 

<sup>1</sup> Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China

<sup>2</sup> College of Computer Science and Technology, Jilin University, Changchun, China  
{shizk14,haoyw16,liangsn17}@mails.jlu.edu.cn, wanli@jlu.edu.cn

<sup>3</sup> The University of Queensland, Brisbane, QLD 4072, Australia  
uqwche12@uq.edu.au

<sup>4</sup> Northeast Normal University, Changchun 130024, China  
yuel031@nenu.edu.cn

**Abstract.** Disease diagnosis can provide crucial information for clinical decisions that influence the outcome in acute serious illness, and this is particularly in the intensive care unit (ICU). However, the central role of diagnosis in clinical practice is challenged by evidence that does not always benefit patients and that factors other than disease are important in determining patient outcome. To streamline the diagnostic process in daily routine and avoid misdiagnoses, in this paper, we proposed a deep multi-source multi-task attention model (DMMAM) for ICU disease diagnosis. DMMAM exploits multi-sources information from various types of complications, clinical measurements, and the medical treatments to support the diagnosis. We evaluate the proposed model with 50 diseases of 9 classifications on an extensive collection of real-world ICU Electronic Health Records (EHR) dataset with 151729 ICU admissions from 46520 patients. Experiments results demonstrate the effectiveness and the robustness of our model.

**Keywords:** Electronic Health Record · Disease prediction · Multi-source multi-task learning · Health care data mining

## 1 Introduction

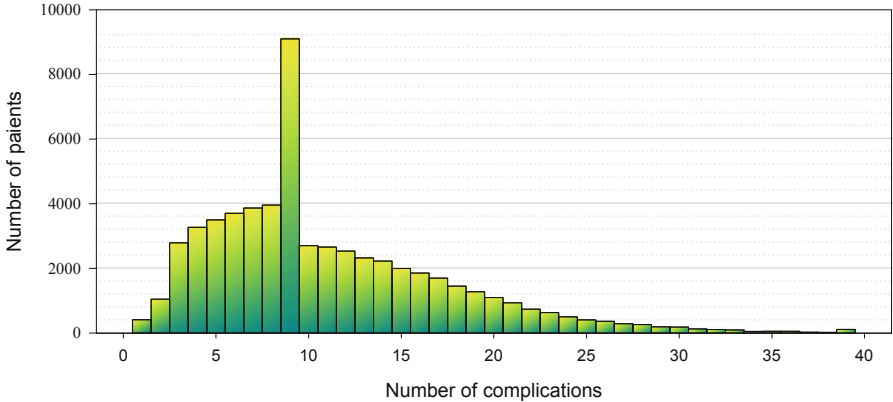
The traditional model of clinical practice incorporates diagnosis, prognosis, and treatment. Diagnosis is fundamental to the practice of medicine and mastery of it is central to the process of both becoming and practicing as a doctor. Moreover, the activity of diagnosis is central to the practice of medicine, and has, to date, received the focused medical and computational science attention which many have argued it warrants [3]. This is beginning to be outburst with an emergent

computer-aided diagnosis, which seeks to explore the activity and its outcomes as a prism through which many issues are played out [14]. It is argued that diagnosis serves many functions for patients, clinicians, and wider society [14], and can be understood both as a category and a process [3]. Diagnosis classifies the sick patient as having or not having a particular disease. Historically, the diagnosis was regarded as the primary guide to treatment and prognosis (“what is likely to happen in the future”), and this is still considered the core component of clinical practice [8].

Intensive care refers to the specialized treatment given to patients who are acutely unwell and require critical medical care. Moreover, an **Intensive Care Unit (ICU)** provides the critical care and life support for acutely ill and injured patients. The ICU is one of the most critically functioning operational environments in a hospital. To healing ICU patients, the clinicians need to actions in a remarkably short period. However, intensivists depend upon a large number of measurements to make daily decisions in the ICU. However, the reliability of these measures may be jeopardized by the effects of therapy [18]. Moreover, in critical illness, what is normal is not necessarily optimal. Diagnosis as the initial step of this medical practice is one of the most important parts of complicated clinical decision making [1].

With **Electronic Health Records (EHR)** growth in biomedical and healthcare communities, it is possible to use bedside computer-aided diagnosis to accurate analysis of medical data, which can greatly benefit the ICU disease diagnosis as well as patient care, and community services. However, the existing work has focused on specialized predictive models that predict a limited set of disease. Such as Long *et al.* use the *IT2FLS* model to diagnosis heart disease [17], Jiri Polivka Jr *et al.* tried to find the mystery of the brain metastatic disease [22], Chaurasia *et al.* [4] use data mining techniques to detect breast cancer and Nilashi *et al.* [20] use neuro-fuzzy technique for hepatitis disease diagnosis. However, the day-to-day clinical practice involves an unscheduled and heterogeneous mix of scenarios and needs different prediction models in the hundreds to thousands [7]. It is impractical to develop and deploy specialized models one by one.

As shown in Fig. 1, this is the complication distribution of patients in the **Medical Information Mart for Intensive Care (MIMIC-III)** [12]. We noticed that the vast majority of patients in the ICU are diagnosed with more than one diseases, that is to say, most of the patients have 5 to 20 complications. Moreover, the human body as organic entities and different systems are closely connected, and no diseases are isolated. In considering this, to establish a single model to diagnosis the majority of the diseases, we designed a multi-source multi-task attention [30] model for ICU diagnosis. The sources refer the different clinical measurements and the medical treatment, and the tasks refer the diagnose of different diseases, the detailed description will in the section of **Problem Definition**. To the best of our knowledge, this is the first time that to utilizing the shared feature space from different disease to boost the diagnose performance.



**Fig. 1.** Complication distribution of patients in MIMIC-III.

The focus of this paper is upon diagnosis as a process, we put the diagnosis into a temporal sequence and treated it as a step-by-step process, in particular from the perspective of the EHR data streaming. We conduct our experiment on real-world MIMIC-III benchmark dataset, and the result shows that our model is highly competitive and outperforms the state-of-the-art traditional methods and commonly used deep learning methods. Furthermore, we evaluated our model on 9 human systems over 50 different kinds of diseases.

The main contributions of this work are summarized as follow:

- **Multiple Perspectives for Disease Formulation.** We formulate ICU disease diagnose as a multi-source and multi-task learning problem, where sources correspond to clinical measurements and medical treatment, tasks correspond to the diagnosis of each disease. This work enables us to use a straightforward model to handle different kinds of diseases over all categories.
- **Diagnosis Step by Step.** For the first time, we treat the disease diagnosis as a gradual process over the observations along the temporal measure and treat sequence as well as the complications.
- **A Novel Integrated Model to diagnose the majority of the disease.** We designed a model DMMAM integrated with the input embedding, window alignment, attention mechanisms, and focal loss functions.
- **Comprehensive Evaluated Experiments.** We conduct experiment on MIMIC-III benchmark dataset on 50 diseases over 9 categories, which covers most of the commonly diseases. The results demonstrate that our method is effective, competitive and can achieve state-of-the-art performance.

The remainder of this paper is organized as follows. We present a review of the recent advances in disease diagnoses briefly in Sect. 2. Section 3 gives out the detailed problem definition and our proposed framework. Section 4 introduced our experiment and our discussions. Section 5 concludes this study with future work.

## 2 Related Work

Diagnosis is the traditional basis for decision-making in clinical practice, inferring the disease from the observations attracts more and more attention in recent years [7, 17, 22, 25, 31]. Existing disease prediction methods can be roughly divided into two categories: clinical based diagnosis [9, 22, 25] and data based diagnosis [7, 17, 31]. Most existing clinical based diagnosis need profound knowledge of medical and most of them are focused on the certain field, such as specific diseases are caused by specific germs [21]. Until the last few years, most of the techniques for computer-aided disease diagnosis were based on traditional machine learning and statistical techniques such as logistic regression, support vector machines (SVM) [27], random forests (RF) [19] and decision tree (DT) [2, 11, 24]. Recently, deep learning techniques have achieved great success in many domains through deep hierarchical feature construction and capturing long-range dependencies in an effective manner [10]. Given the rise in popularity of deep learning approaches and the increasingly vast amount of clinical electronic data, there has also been an increase in the number of publications applying deep learning to diseases diagnosis tasks [5–7, 20] which yield better performance than traditional methods and require less time-consuming preprocessing and feature engineering. For instance, Zhenping *et al.* [5] use the Best Mimic Model for ICU outcome prediction and got average 0.1 Area under Receiver Operating Characteristic (AUROC) score than SVM, LR and DT, Zachary C *et al.* learned to diagnose with long short-term memory (LSTM) recurrent neural networks and got average 0.5981 F1 scores over 6 different diseases.

However, all these methods are designed for a specific disease based on either the intensive use of domain-specific knowledge or taking advantage of advanced statistical methods. Specifically, studies have been conducted on Alzheimer’s disease [31], heart disease [17], chronic kidney disease [28], and abdominal aortic aneurysm [13]. Moreover, these models have been developed to anticipate needs and focused on specialized predictive models that predict a limited set of diseases. However, the day-to-day clinical practice involves an unscheduled and heterogeneous mix of scenarios and needs different prediction models in the hundreds to thousands. It is impractical to develop and deploy specialized models one by one [7]. So it is significant to develop a unified model and can apply for the majority of diseases. This is beautiful dovetails to the multi-task learning, each disease can be treated as a single learning task. Note that many approaches to multi-task learning (ML) in the literature deal with a similar setting: They assume that all tasks are associated with the single output, e.g., the multi-class MNIST dataset is typically cast as 10 binary classification tasks. More recent approaches deal with a more realistic, heterogeneous setting where each task corresponds to a unique set of output [23]. We can not simply apply their approaches to ours, because we multiple clinical observations, multiple, and multiple medical treatments cannot be integrated into the existing frameworks.

More importantly, the human body as organic entities and different systems are intimately connected, and no diseases are isolated, so there may be little

difference between the complications. Therefore, based on our experiments it is hard for traditional methods to apply to such huge dataset over 50 kinds of diseases.

Inspired by the above problems, in this paper, we propose a general methodology, namely Deep Multi-source Multi-task Attention Model (DMMAM), to predict the disease from multi-modal data jointly. Here the sources indicate the clinical measurements and the medical treatments, the tasks represent the diagnosis of the diseases. In our work, the variables include not only the continuous clinical variables for regression (time series step by step regression) but also the categorical variable for classification (i.e., the class label for diseases classification). We treat the estimation of different diseases as different tasks, and multi-task learning [31] method developed in the machine learning community for joint learning. Multi-task learning can effectively increase the sample size that we are using to train the model because the samples of some kinds of disease are really small, which are not enough for learning (see Table 1). Specifically, at first, we assume that related tasks share a common relevant feature subset such as the age, temperature, heartbeat, blood pressure, *et al.* but with a varying amount of influence on each task, and thus adopt a hand engineered feature selection method to abstain a common feature subset for different tasks simultaneously. Then, we use a window alignment to adjust the time window between different sources and use one dense layer to reduce the dimensionality. Besides, we use two attention layer to capture the correlations between the different input sources as well as each time step. Finally, we use a gated recurrent unit (GRU) to fuse the above-selected features from each modality to estimate multiple regression and classification variables.

We will detail the problem definition in Sect. 3 and our proposed method in Sect. 4.

### 3 Proposed Framework

#### 3.1 Problem Statement

For a given ICU stay length of  $T$  hours, and a collection of diagnostic results  $R_t, t \in T$ , it is assumed that we have a series of clinical observation:

$$O(t) = \begin{cases} R_t, & \text{if } R_t \notin \emptyset \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

where  $O(t)$  is vector of bedside observations at time  $t$ .  $O(t) = P_a^i \Theta Q_b^j$ , where  $P_a^i$  represent the  $i$ -th clinical measurement at time  $a$ ,  $Q_b^j$  represent the  $j$ -th medical treatment at time  $b$ , and  $\Theta$  is a window alignment operation between  $P_a^i$  and  $Q_b^j$ , and  $R_t$  represent the diagnostic result at time  $t$ . Our objective is to generate a sequence-level disease prediction at each sequence step. The type of prediction depends on the specific task and can be donated as a discrete scalar vector  $R_t^i$  for the multi-task classification. As all tasks are at least somewhat noisy, when

training a model  $Task_i$ , we expect to learn a good representation for  $Task_i$  that ideally ignore the data-dependent noise and generalize well. By sharing representations between related tasks, we can enable our model to generalize better on our original task.

### 3.2 Multi-modal Multi-task Temporal Learning Framework for Temporal Data

Inspired by Daoqiang Zhang and Dinggang Shen’s work [31], we treat the diagnosis of the diseases as a sequential multi-modal multi-task (SM3T) learning problem. The multi-modal represents the clinical measurements and the medical treatments. The tasks represent the diagnosis. The framework can simultaneously learn multiple tasks from multi-model temporal data. Figure 2 illustrates the proposed SM3T method and a comparison with the existing learning methods.

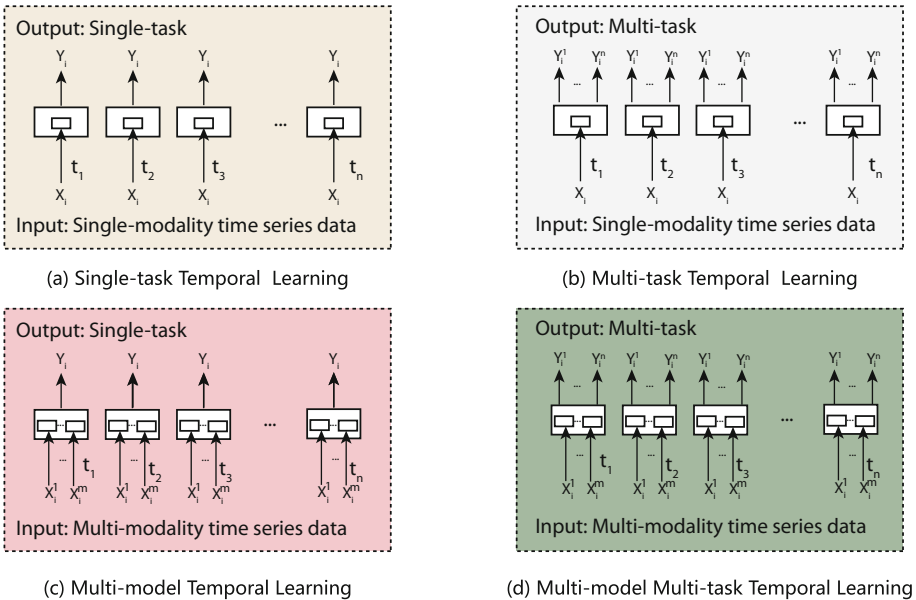


Fig. 2. Multi-modal multi-task temporal learning framework for temporal data.

Figure 2(a) is single-modality single-task temporal learning, each subject has only one modality of data represented as  $x_i$  at each time step, and each subject corresponds to only one task denoted as  $Y_i$ , this is the most commonly used learning method; Fig. 2(b) is single-modality multi-task temporal learning the input is similar as single-task temporal learning, but each object corresponds to multiple tasks denoted as  $Y_i^1, Y_i^2, Y_i^3, \dots, Y_i^n, n > 1$ ; Fig. 2(c) is multi-modality single-task temporal learning, each subject has multiple modalities of data represented as

$x_i^1, x_i^2, x_i^3, \dots, x_i^n, n > 1$  at each time step and each subject corresponds to only one task denoted as  $Y_i$ ; Fig. 2(d) is multi-modality multi-task temporal learning, each subject has multiple modality of data represented as  $x_i^1, x_i^2, x_i^3, \dots, x_i^n, n > 1$  at each time step and each subject corresponds to multiple tasks denoted as  $Y_i^1, Y_i^2, Y_i^3, \dots, Y_i^n, n > 1$ .

Similar to Zhang's *et al.* [31] we can formally define the SM3T learning as below. Given  $N$  training subjects over  $T$  time span and each is having  $M$  modalities of data, represented as:

$$x_i^t = \{x_i^t(1), x_i^t(2), \dots, x_i^t(m), \dots, x_i^t(M)\}, i = 1, 2, \dots, N \quad (2)$$

our SM3T method jointly learns a series of models corresponding to  $Y$  different tasks denoted as:

$$Y_i = \{y_i^t(1), y_i^t(2), \dots, y_i^t(j), \dots, y_i^t(Y)\}, j = 1, 2, \dots, N \quad (3)$$

Noting that SM3T is a general learning framework, and here we implement it through an attention framework as shown in Fig. 3. The x-axis represents the sequential data stream at time  $t$ , the y-axis represents the actions conducted on each  $t$  point and z-axis is the modalities of the input sources. In our experiment,  $N = 2$  (e.g.,  $S1 =$  clinical measurements and  $S2 =$  medical treatment) are used for jointly learning models corresponding to different tasks. We will detail the inner action of the SM3T framework in the following sections.

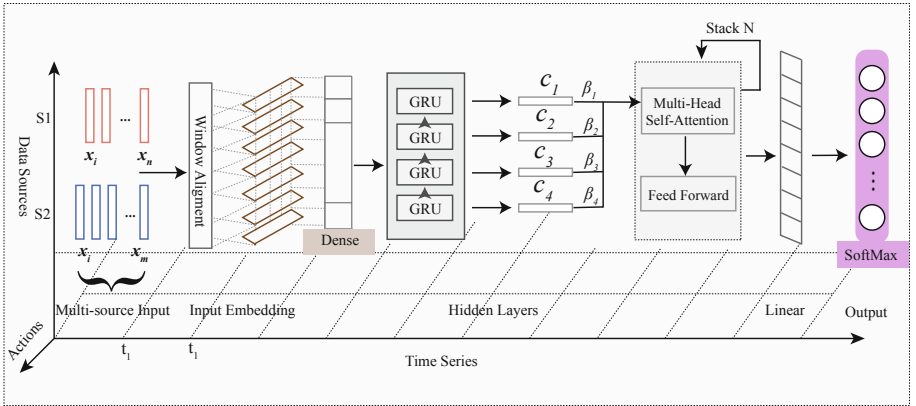


Fig. 3. The proposed multi-source multi-task attention model.

### 3.3 Input Embedding and Window Alignment

Give the  $R$  actions for each step for each step  $t$ , the first step in our model is to generate an embedding that captures the dependencies across different disease without the temporal information. In the embedding step, let  $N$  denote the number of diseases. The diagnosis process is first designed for each disease



without temporal information. Let  $P$  denote the ICU patients. The  $p$ -th patient have  $h$  diagnosis results at time  $t$ , and  $p$ -th patients with  $h$ -th diseases is associated with two feature vectors  $Sa_p^h(t)$  and  $Sb_p^h(t)$  derived from the EHR, where  $Sa_p^h(t)$  donate the clinical measurements and  $Sb_p^h(t)$  donates the medical treatments. The dimension of  $Sa$  and  $Sb$  are  $\alpha$  and  $\beta$ , respectively. Combined  $Sa$  and  $Sb$ , we generated a new feature vector  $\Phi^h$  for the  $p$ -th patient:

$$\Phi^p \equiv [\phi_1^p(t), \phi_2^p(t), \dots, \phi_h^p(t)] \quad (4)$$

$$\phi_p^h(t) = \lambda_1^h Sa_p^h(t) \star \lambda_2^h Sb_p^h(t) \quad (5)$$

where  $\star$  is **Window Alignment** operation, and  $\lambda_1$  and  $\lambda_2$  are trainable hyper-parameters for each disease.

Since our framework contains multiple actions, medical treatments  $Sb$  and clinical measurements  $Sa$ . The intentions of why we add a window alignment operation is that according to the common medical sense, the effect of treatment usually has some delay to the measurements. Assume  $Sa_p^h(ti)$  represent the clinical measurements at time  $ti$  and  $Sa_p^h(tj)$  represent the medical treatments at time step  $tj$ . The alignment is performed by mapping  $Sa_p^h(ti)$  and  $Sa_p^h(tj)$  into a unique time step  $S_p^h(t)$ . The alignment parameters  $\lambda_i^h$  are learned according to the patients and disease respectively. We found that  $tj$  usually later than  $ti$ , and this well accords with the prevailing medical sense.

### 3.4 Dense Layer

To balance the computational cost as well as the predictable performance, we need to reduce the dimensions before we transfer the raw medical data to the next process step. The typical way is to concatenate an embedding at every step in the sequence. However, due to the high-dimensional of the clinical features, ‘‘cursed’’ representation which is not suitable for learning and inference. Inspired by the Trask’s work [29] in Natural Language Processing (NLP) and Song’s [26] in clinical data processing, we add a dense layer to unify and flatten the input features. To prevent overfitting, we set dropout = 0.38 here.

### 3.5 The Gated Recurrent Unit Layer

The gated recurrent unit layer (GRU) takes the sequence of action  $\{x_t\}_{t \geq 1}^T$  from the previous dense layer and then associate  $p$ -th patient with a class label vector  $Y$  along with the time span, donates the class label for the  $p$ -th patient with the  $n$ -th disease at time  $T$ .  $Y_p^n(t)$  is set ass follows:

$$Y_p^n(t) = \begin{cases} diseaseID, & \text{if diagnosis recorded at time } t \\ 0, & \text{otherwise.} \end{cases} \quad (6)$$

We create a  $T$ -dimensional response vector for the  $p$ -th patient:

$$Y^{(p)} = (y_{p,1}, y_{p,2}, \dots, y_{p,p_t})^\top \quad (7)$$

For the diagnosis of ICU patients, we adopted GRU and represent the posterior probability of the outcome of patient  $p$  has  $y - th$  disease as:

$$Pr[P_y^n(t) = 1 | \phi_h^p(t)] = \sigma(\omega^{(p)T} \phi_h^p(t)) \quad (8)$$

where  $\phi(a)$  is the sigmoid function  $\sigma(a) \equiv (1 + \exp(-a))^{-1}$  and  $\omega^{(p)}$  is a  $\alpha + \beta$  dimensional model parameter vector for the  $p - th$  patient.

To learn the mutual information of data resulting from the customization, we model for all disease jointly, so that we can share the same vector space across the disease, this is very useful for those diseases with fewer samples. We represent the trainable parameters of the GRU as a  $(Sa + Sb) \times T$   $W \equiv [\omega^1, \omega^2, \dots, \omega^t]$ .

### 3.6 Multi-head Attention and Feed Forward

This attention layer is designed to capture the dependencies of the whole sequence, as we treated the diagnosis as a step-by-step process. In the ICU scenario, the actions (clinical measurements and medical treatments) closer to the current position are critical in helping the diagnosis. However, the observations further are less critical. Therefore, we should consider information entropy differently based on the positions which we make observations.

Inspired by [30], we use H-heads attention to create multiple attention graphs, and the resulting weighted representations are concatenated and linearly projected to obtain the final representation. Moreover, we also add 1D convolutional sub-layers with kernel size 2. Internally, we use two of these 1D convolutional sub-layers with ReLU (rectified linear unit) activation in between. Residue connections are used in these sub-layers. Unlike the previous work [1, 4, 7, 11] making the diagnosis only once after a specific timestamp, we give out prediction at each timestamp. This is because the diagnosis results may change during the ICU stay and we make it as a dynamic procedure. This is more helpful for the ICU clinicians because they need to know the patients' possible disease at any time other than at the particular time. We stack the attention module N times and using the final representations in the final model. Moreover, this attention layer is task wise, that is to say if this attention will only work when this attention is helpful to the diagnosis.

### 3.7 Linear and Softmax Layers

The linear layer is designed to obtain the logits from the unified output of attention layer. The activation function used in this layer is ReLU. The last layer is preparing for the output based on different tasks. We use softmax to classify the different diseases, and the loss function is:

$$Loss_d = \frac{1}{N} \sum_{n=1}^N -(y_k \bullet \log(\bar{y}_k) + (1 - y_k)). \quad (9)$$

where N donate the number of diseases. Due to the distribution of the training set we also introduce Focal Loss as our loss function [16].

**Table 1.** Description of the prediction tasks based on ICD 9 code.

Category	ICD 9	Title	SampleSize	Age	
<b>1: Infectious and Parasitic</b>	008.45	Int inf clstridium dfcile	2672	69.07 ± 24.31	
	038.9	Unspecified septicemia	5787	69.11 ± 32.13	
<b>2: Neoplasms</b>	197.0	Secondary malig neo lung	866	62.23 ± 13.31	
	197.7	Second malig neo liver	926	64.63 ± 17.47	
	198.5	Secondary malig neo bone	984	63.59 ± 12.77	
<b>3: Endocrine, Nutritional, Metabolic and Immunity</b>	250.00	DMII wo cmp nt st uncuntr	10585	71.40 ± 28.41	
	250.40	DMII renl nt st uncuntrld	1574	69.26 ± 20.04	
	250.60	DMII neuro nt st uncntrl	1793	70.02 ± 26.25	
	263.9	Protein-cal malnutr NOS	2258	65.95 ± 26.35	
<b>4: Blood and Blood-forming Organs</b>	280.0	Chr blood loss anemia	1346	68.34 ± 25.88	
	280.9	Iron defic anemia NOS	1992	67.38 ± 39.21	
	285.1	Ac posthemorrhag anemia	6998	69.10 ± 36.81	
	285.21	Anemia in chr kidney dis	2616	66.70 ± 28.35	
	285.29	Anemia-other chronic dis	2225	67.45 ± 32.21	
	285.9	Anemia NOS	8253	67.90 ± 34.13	
<b>5: Circulatory System</b>	397.0	Tricuspid valve disease	1286	77.26 ± 40.76	
	401.9	Hypertension NOS	23153	71.27 ± 32.66	
	403.90	Hy kid NOS w cr kid I-IV	4712	81.32 ± 45.61	
	403.91	Hyp kid NOS w cr kid V	3756	65.27 ± 19.49	
	410.71	Subendo infarct initial	4474	74.17 ± 30.51	
	411.1	Intermed coronary synd	2200	69.42 ± 22.56	
	412	Iron defic anemia NOS	4479	74.93 ± 36.99	
	413.9	Angina pectoris NEC/NOS	1468	70.64 ± 27.84	
	414.00	Crrny athrslc natve vssl	2415	78.53 ± 37.30	
	414.01	Cor ath unsp vsl ntv/gft	14585	73.24 ± 32.09	
	414.8	Chr ischemic hrt dis NEC	1526	74.54 ± 28.52	
	431	Intracerebral hemorrhage	1561	69.71 ± 28.83	
	433.10	Ocl crtd art wo infrct	1109	75.77 ± 30.39	
	434.91	Crbl art ocl NOS w infrc	907	69.41 ± 28.22	
	<b>6: Respiratory System</b>	482.41	Meth sus pneum d/t Staph	1297	64.56 ± 22.81
		486	Pneumonia organism NOS	7779	68.51 ± 32.89
491.21		Obs chr bronc w(ac) exac	1851	72.91 ± 24.79	
493.20		Asthma NOS	1215	69.22 ± 26.13	
493.90		Chronic obst asthma NOS	2781	59.18 ± 30.16	
<b>7: Digestive System</b>	571.2	Cirrhosis of liver NOS	1529	55.93 ± 12.54	
	571.5	Alcohol cirrhosis liver	1820	60.29 ± 16.73	
<b>8: Genitourinary System</b>	584.5	Ac kidney fail tubr necr	3567	65.98 ± 24.11	
	584.9	Acute kidney failure NOS	3564	71.45 ± 36.21	
	585.6	Chronic kidney dis NOS	2720	62.39 ± 20.38	
	585.9	End stage renal disease	4942	79.01 ± 41.90	
	600.00	BPH w/o urinary obs/LUTS	1850	79.81 ± 35.58	
<b>9: Conditions originating in the perinatal period</b>	765.18	Preterm NEC 2000-2499g	621	0.03 ± 0.03	
	765.19	33-34 comp wks gestation	557	0.02 ± 0.02	
	765.27	35-36 comp wks gestation	545	0.04 ± 0.03	
	765.28	Preterm NEC 2500+g	642	0.02 ± 0.02	
	769	Respiratory distress syn	511	0.10 ± 0.09	
	770.6	Primary apnea of newborn	535	0.02 ± 0.03	
	770.81	NB transitory tachypnea	331	0.10 ± 0.08	
	774.2	Neonat jaund preterm del	1021	0.08 ± 0.08	
774.6	Fetal/neonatal jaund NOS	514	0.02 ± 0.04		

## 4 Experiment

### 4.1 Data Description

We use a real-world dataset from MIMIC III<sup>1</sup> to evaluate our proposed approach. MIMIC-III is a large, publicly-available database comprising de-identified health-related data associated with approximately sixty thousand admissions of patients who stayed in critical care units of the Beth Israel Deaconess Medical Center between 2001 and 2012. The open nature of the data allows clinical studies to be reproduced and improved in ways that would not otherwise be possible [12]. In our experiment, we treat each ICU stay as a single case, because different ICU stay from the same patient may have diagnosed with a different disease. Moreover, this operation can help us to obtain more samples to train. As shown in Table 1, this is the first time that disease diagnosis conduct on such huge amount categories. We category the dataset based on the International Classification of Diseases (ICD) code, ICD-9, and we select 151729 ICU admissions over 50 commonly diagnosed disease. As shown in Fig. 1, most patients have multiple complications, and we collected all the complications in the whole ICU process temporally. Unlike the previous work, we did not filter any patients, this may results low performance, compared with related work. For the features, we included 529 clinical measurements features and 330 medical treatment features. Due to the abundant and mussy training samples, the performance between different disease is hugely different.

### 4.2 Experiment Settings

Our experiment includes over 40000 patients among 9 categories of 50 kinds of disease, the ICD9 code range from 001 to 779. A measure of the diagnosed disease, we set the outcome is “true” if the prediction result is right between the diagnose time span we observed the disease otherwise “false”. In the training process, we will give out predict every time step only if there are observations during this time step, but in the test process we can give out diagnosis at every time step, and the time span can be customized. The learning rate in this experiment is 0.001, and the epochs size is 30. In our experiment, we set the batch size to 32, with ADAM optimizer and set dropout = 0.35. According to our experiment, we can get most of the best performance when then attention stack for 4 times. In order to conduct all the experiment in the same data, we manually divide the training set, validation set, and test set, we listed it in the Table 2.

### 4.3 Compared Methods

We compared our proposed method with 6 commonly used methods, i.e., Logistic Regression (LR) with L2 regularization, Random Forest (RF), Support Vector Machine (SVM), Decision Tree (DT), GRU, and the-state-of-the-art LSTM

---

<sup>1</sup> Data available at <https://mimic.physionet.org/>.

**Table 2.** Experiment settings for training, validating and test.

Task	Train	Validation	Test	Task	Train	Validation	Test
008.45	1870	534	268	414.8	1068	305	153
038.9	4050	1157	580	431	1092	312	157
197.0	606	173	87	433.10	776	221	112
197.7	648	185	93	434.91	634	181	92
198.5	688	196	100	482.41	907	259	131
250.00	7409	2117	1059	486	5445	1555	779
250.40	1101	314	159	491.21	1295	370	186
250.60	1255	358	180	493.20	850	243	122
263.9	1580	451	227	493.90	1946	556	279
280.0	942	269	135	571.2	1070	305	154
280.9	1394	398	200	571.5	1274	364	182
285.1	4898	1399	701	584.5	2496	713	358
285.21	1831	523	262	584.9	2494	712	358
285.29	1557	445	223	585.6	1904	544	272
285.9	5777	1650	826	585.9	3459	988	495
397.0	900	257	129	600.00	1295	370	185
401.9	16207	4630	2316	765.18	434	124	63
403.90	3298	942	472	765.19	389	111	57
403.91	2629	751	376	765.27	381	109	55
410.71	3131	894	449	765.28	449	128	65
411.1	1540	440	220	769	357	102	52
412	3135	895	449	770.6	374	107	54
413.9	1027	293	148	770.81	231	66	34
414.00	1690	483	242	774.2	714	204	103
414.01	10209	2917	1459	774.6	359	102	53

based method [15]. Due to the page limitation we only listed the two of the top two best methods in our paper. The first one is Logistic Regression (LR) with L2 regularization, and the second is the-state-of-the-art LSTM based method we listed LSTM+ in Table 3. As mentioned above, to ensure every evaluation method uses the same data, we fixed the dataset. As shown in Table 2 the validation and test data we use is approximately 25% of the whole dataset.

**Table 3.** Performance evaluation on each diagnose task.

Cat.	Task	LR			LSTM+			DMMAM(our method)		
		F1	Acc	Recall	F1	Acc	Recall	F1	Acc	Recall
1	008.45	0.5822	0.6639	0.4784	0.8123	0.6840	<b>1</b>	<b>0.8641</b>	<b>0.7240</b>	<b>1</b>
	038.9	0.5822	0.6639	0.9345	0.7442	0.5259	<b>1</b>	<b>0.8641</b>	<b>0.8171</b>	0.9216
2	197.0	0.5593	0.5679	0.2414	0.7919	0.5392	0.7122	<b>0.8515</b>	<b>0.8281</b>	<b>0.8846</b>
	197.7	0.5895	0.5964	0.3333	0.7570	0.6357	0.8663	<b>0.8162</b>	<b>0.7172</b>	<b>0.8945</b>
	198.5	0.5366	0.5286	0.4500	0.6012	0.5214	0.5667	<b>0.6842</b>	<b>0.7118</b>	<b>0.6457</b>
3	250.00	0.5465	0.6546	<b>0.9754</b>	0.5443	0.6533	0.0531	<b>0.6545</b>	<b>0.7101</b>	0.6153
	250.40	0.8549	0.8941	0.0189	<b>0.9485</b>	<b>0.9021</b>	<b>1.0000</b>	<b>0.9485</b>	<b>0.9021</b>	<b>1.0000</b>
	250.60	0.8382	0.8892	0.0056	0.9413	0.8892	0.9000	<b>0.9613</b>	<b>0.9292</b>	<b>1.0000</b>
	263.9	0.8135	0.8602	0.0752	<b>0.9252</b>	<b>0.8608</b>	<b>1.0000</b>	<b>0.9252</b>	<b>0.8608</b>	<b>1.0000</b>
4	280.0	0.9139	0.9412	0.67454	0.6364	0.4116	0.5483	<b>0.9704</b>	<b>0.9425</b>	<b>1.0000</b>
	280.9	0.8740	0.9130	0.5050	0.9557	0.9152	0.9210	<b>0.9755</b>	<b>0.9347</b>	<b>1.0000</b>
	285.1	0.6405	0.6398	0.4037	0.8204	0.6995	0.9897	<b>0.8482</b>	<b>0.7412</b>	<b>0.9988</b>
	285.21	0.8531	0.8717	0.1832	0.9409	0.8883	0.8328	<b>0.9709</b>	<b>0.9283</b>	<b>1.0000</b>
	285.29	0.8589	0.9024	0.4327	<b>0.9503</b>	<b>0.9054</b>	0.9200	<b>0.9503</b>	<b>0.9054</b>	<b>1.0000</b>
	285.9	0.5216	0.5196	0.7167	0.5852	0.4996	0.5447	<b>0.7492</b>	<b>0.5533</b>	<b>0.8803</b>
5	397.0	0.8429	0.8587	0.1163	0.9453	0.8962	0.9991	<b>0.9457</b>	<b>0.8970</b>	<b>1.0000</b>
	401.9	0.5830	0.6232	0.1354	0.7277	0.6137	0.2380	<b>0.9213</b>	<b>0.9137</b>	<b>0.7612</b>
	403.90	0.6787	0.6621	0.6271	0.8065	0.6838	0.9027	<b>0.8255</b>	<b>0.7079</b>	<b>0.9466</b>
	403.91	0.7841	0.7892	0.4495	0.8777	0.7824	0.9956	<b>0.8795</b>	<b>0.7852</b>	<b>0.9993</b>
	410.71	0.7007	0.7062	0.3764	0.8293	0.7159	0.9314	<b>0.8333</b>	<b>0.7776</b>	<b>0.9499</b>
	411.1	0.7835	0.7670	0.5818	0.8559	0.7534	0.8887	<b>0.8705</b>	<b>0.7749</b>	<b>0.9177</b>
	412	0.6930	0.7131	0.2806	0.8122	0.6924	0.8951	<b>0.8382</b>	<b>0.7228</b>	<b>0.9668</b>
	413.9	0.8326	0.8611	0.1014	0.9345	0.8771	0.9937	<b>0.9376</b>	<b>0.8827</b>	<b>1.0000</b>
	414.00	0.6858	0.6536	0.5331	0.7307	0.6081	0.6588	<b>0.7419</b>	<b>0.6129</b>	<b>0.6894</b>
	414.01	0.5830	0.6232	0.2503	0.7606	0.6137	<b>1.0000</b>	<b>0.8508</b>	<b>0.7091</b>	<b>1.0000</b>
	414.8	0.8215	0.8468	0.1046	0.9327	0.8739	0.9955	<b>0.9350</b>	<b>0.8779</b>	<b>1.0000</b>
	431	0.8619	0.8540	0.5669	0.9317	0.8723	0.9954	<b>0.9332</b>	<b>0.8747</b>	<b>1.0000</b>
	433.10	0.8588	0.8899	0.0089	<b>0.9532</b>	<b>0.9106</b>	<b>1.0000</b>	<b>0.9532</b>	<b>0.9106</b>	<b>1.0000</b>
	434.91	0.8873	0.9058	0.0652	0.9123	0.9074	0.8975	<b>0.9619</b>	<b>0.9266</b>	<b>1.0000</b>
	6	482.41	0.8705	0.9071	0.0153	0.9542	0.5091	0.8320	<b>0.9762</b>	<b>0.7210</b>
486		0.4328	0.5337	0.9345	0.6016	0.5210	0.0292	<b>0.9542</b>	<b>0.9125</b>	<b>1.0000</b>
491.21		0.8180	0.8651	0.0269	<b>0.9338</b>	<b>0.8758</b>	<b>1.0000</b>	<b>0.9338</b>	<b>0.8758</b>	<b>1.0000</b>
493.20		0.8782	0.9158	0.8861	0.9575	0.9185	0.8921	<b>0.9775</b>	<b>0.9432</b>	<b>1.0000</b>
493.90		0.7508	0.7989	0.1039	0.8972	0.8136	0.8020	<b>0.9454</b>	<b>0.8732</b>	<b>1.0000</b>
7	571.2	0.5626	0.5685	0.4416	0.6866	0.5625	0.8846	<b>0.6951</b>	<b>0.5744</b>	<b>0.8956</b>
	571.5	0.5626	0.5685	0.6758	0.4111	0.5625	0.3377	<b>0.7204</b>	<b>0.6744</b>	<b>0.7455</b>
8	584.5	0.7662	0.7505	0.2817	0.9257	0.8618	<b>1.0000</b>	<b>0.9259</b>	<b>0.8620</b>	<b>1.0000</b>
	584.9	0.4235	0.5251	0.1003	0.5016	0.4945	0.0545	<b>0.7739</b>	<b>0.7173</b>	<b>1.0000</b>
	585.6	0.8768	0.8966	0.2169	0.9441	0.8941	1.0000	<b>0.9642</b>	<b>0.8943</b>	<b>1.0000</b>
	585.9	0.4858	0.4473	0.7859	0.8933	0.8072	<b>0.7892</b>	<b>0.9241</b>	<b>0.8124</b>	<b>0.8600</b>
	600.00	0.8984	0.9184	0.0919	<b>0.9637</b>	<b>0.9299</b>	<b>1.0000</b>	0.9627	0.9281	<b>1.0000</b>
	600.00	0.8984	0.9184	0.0919	<b>0.9637</b>	<b>0.9299</b>	<b>1.0000</b>	0.9627	0.9281	<b>1.0000</b>
9	765.18	0.8260	0.8555	0.0794	0.9361	0.8799	0.9979	<b>0.9372</b>	<b>0.8818</b>	<b>1.0000</b>
	765.19	0.8272	0.8593	0.5420	0.8446	0.8249	0.8213	<b>0.9357</b>	<b>0.8799</b>	<b>0.9769</b>
	765.27	0.8499	0.8518	0.2545	0.9446	0.8949	0.9979	<b>0.9456</b>	<b>0.8968</b>	<b>1.0000</b>
	765.28	0.8225	0.8255	0.2462	0.9340	0.8762	0.9979	<b>0.9359</b>	<b>0.8799</b>	<b>1.0000</b>
	769	0.8401	0.8349	0.2308	<b>0.9487</b>	<b>0.9024</b>	<b>1.0000</b>	<b>0.9487</b>	<b>0.9024</b>	<b>1.0000</b>
	770.6	0.8568	0.8518	0.3519	0.9476	0.9006	0.9192	<b>0.9486</b>	<b>0.9238</b>	<b>1.0000</b>
	770.81	0.9044	0.9343	0.9101	0.9680	0.9381	1.0000	<b>0.9691</b>	<b>0.9392</b>	<b>1.0000</b>
	774.2	0.6710	0.6379	0.4608	0.6174	0.4953	0.5035	<b>0.7904</b>	<b>0.6717</b>	<b>0.7657</b>
	774.6	0.8679	0.8593	0.4423	0.9497	0.9043	0.9102	<b>0.9765</b>	<b>0.9343</b>	<b>1.0000</b>

#### 4.4 Evaluation Metric

To provide a comparison among the mentioned techniques, three evaluation techniques were used in this research: F1-Measure, Accuracy, and Recall. Those evaluation techniques are defined as:

$$\text{Accuracy} = \frac{TF + TN}{TP + FP + TN + FN} \quad \text{Recall} = \frac{TP}{TP + FN} \quad (10)$$

$$\text{F1-Measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (11)$$

where  $TP$  and  $FP$  are the number of true positive and false negative, respectively.

#### 4.5 Experiment Results and Discussions

Table 3 shows the prediction results. We can see that our model is significantly outperformed than all the baseline methods. Because we did not filter any ICU admissions and included all categories of the disease, so some evaluation metrics of our experiment are lower than those results appeared in Chen *et al.*'s work [15] (marked as LSTM+ in Table 3), but under the same experiment settings, our can always achieved the best performance. We can see that the number of the sample can greatly improve the diagnosis performance, the more samples, the better performance can achieve.

We discovered that the difference among categories are more evident than the diseases in the same category, and can pass average 3.2% in accuracy. The disease in category 3, Endocrines, Nutritional, Metabolic and Immunity is the hardest disease to diagnosis in our model, and the disease of Conditions originating in the perinatal period in category 9 are the easiest ones to diagnosis. This is because there are greater diversities between category 9 and others, and there are smaller diversities between category 3 and others. Besides, the disease in the same categories have different diagnosis performance indicate that there is a higher relevance in the same system. We also conducted the ablation studies on the process of diagnosis, and the results show that the multi-source and multi-task can help us improved the performance among all the tasks over 3.6 percent in F1 scores. That is to say, by share the context feature space in the hidden layers the DMMAM can significantly improve the performance.

## 5 Conclusion and Future Work

In this study, we presented a new model named DMMAM for the disease diagnosis in the circumstances of the ICU. We modeled the ICU disease diagnosis as a multi-source multi-task classification problem. Moreover, we treat the diagnosis as a gradually process along the clinical measurements and the clinical treatments. The significances of our proposed model can be identified as:

1. **We considered the diversity of complications.** This both accords with the medical situation that no disease is isolated and different diseases have different diagnostic criteria and different treatment methods, the proposed multi-source multi-task model can perfectly suitable for this situations;
2. **We considered the diagnosis sequential relationship.** By introducing the attention layer we simulated the clinicians' diagnosis process and captured the interaction information among the sequence.
3. **Solved the imbalance problem.** The sample variance among the training data is hugely among different diseases. For example, the unspecified essential hypertension has 23153 samples. However, the secondary malignant neoplasm of the lung has only 866 samples. So if we are learning diagnosis without any precautionary measures, the diagnosis result would definitely to the majority ones. By using focal loss function, we alleviated problem caused by the unbalance of the dataset in the training process.

We conducted our experiment on 50 diseases over 167884 samples the results show the robustness and high accuracy. Moreover, this is the first time that diagnosis been conducted on such huge dataset. Nevertheless, how to use these diagnoses in further clinical actions remains a challenge in scientific research, and future work can be focused on this problem.

**Acknowledgement.** This work was supported by the Nature Science Foundation of Jilin Province (20180101330JC, 20190302029GX), the Fundamental Research Funds for the Central Universities (No. 2412017QD028), the China Postdoctoral Science Foundation (No. 2017M621192), the Scientific and Technological Development Program of Jilin Province (No. 20180520022JH, 20190302109GX). The authors also gratefully acknowledge the financial support from China Scholarship Council (No. 201706170617).

## References

1. Ahmadi, H., Gholamzadeh, M., Shahmoradi, L., Nilashi, M., Rashvand, P.: Diseases diagnosis using fuzzy logic methods: a systematic and meta-analysis review. *Comput. Methods Programs Biomed.* **161**, 145 (2018)
2. Azar, A.T., El-Metwally, S.M.: Decision tree classifiers for automated medical diagnosis. *Neural Comput. Appl.* **23**(7–8), 2387–2403 (2013)
3. Blaxter, M.: Diagnosis as category and process: the case of alcoholism. *Soc. Sci. Med. Part A Med. Psychol. Med. Sociol.* **12**, 9–17 (1978)
4. Chaurasia, V., Pal, S.: A novel approach for breast cancer detection using data mining techniques (2017)
5. Che, Z., Purushotham, S., Khemani, R., Liu, Y.: Interpretable deep models for ICU outcome prediction. In: *AMIA Annual Symposium Proceedings*, vol. 2016, p. 371. American Medical Informatics Association (2016)
6. Chen, M., Hao, Y., Hwang, K., Wang, L., Wang, L.: Disease prediction by machine learning over big data from healthcare communities. *IEEE Access* **5**, 8869–8879 (2017)
7. Choi, E., Bahadori, M.T., Schuetz, A., Stewart, W.F., Sun, J.: Doctor AI: predicting clinical events via recurrent neural networks. In: *Machine Learning for Healthcare Conference*, pp. 301–318 (2016)



8. Del Mar, C., Doust, J., Glasziou, P.: Clinical thinking; evidence, communication and decision-making (2006)
9. Detemmerman, L., Olivier, S., Bours, V., Boemer, F.: Innovative PCR without dna extraction for African sickle cell disease diagnosis. *Hematology* **23**(3), 181–186 (2018)
10. Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y.: *Deep Learning*, vol. 1. MIT Press, Cambridge (2016)
11. Hao, Y., Zuo, W., Shi, Z., Yue, L., Xue, S., He, F.: Prognosis of thyroid disease using MS-apriori improved decision tree. In: Liu, W., Giunchiglia, F., Yang, B. (eds.) *KSEM 2018. LNCS (LNAI)*, vol. 11061, pp. 452–460. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-99365-2\\_40](https://doi.org/10.1007/978-3-319-99365-2_40)
12. Johnson, A.E., et al.: Mimic-III, a freely accessible critical care database. *Sci. Data* **3**, 160035 (2016)
13. Johnson, M.J., Willsky, A.S.: Bayesian nonparametric hidden semi-Markov models. *J. Mach. Learn. Res.* **14**, 673–701 (2013)
14. Jutel, A., Nettleton, S., et al.: Towards a sociology of diagnosis: reflections and opportunities. *Soc. Sci. Med.* **73**(6), 793–800 (2011)
15. Lin, C., et al.: Early diagnosis and prediction of sepsis shock by combining static and dynamic information using convolutional-LSTM. In: 2018 IEEE International Conference on Healthcare Informatics (ICHI), pp. 219–228. IEEE (2018)
16. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988 (2017)
17. Long, N.C., Meesad, P., Unger, H.: A highly accurate firefly based algorithm for heart disease prediction. *Expert Syst. Appl.* **42**(21), 8221–8231 (2015)
18. Marshall, J.C.: Measurements in the intensive care unit: what do they mean? *Crit. Care* **7**(6), 415 (2003)
19. Nguyen, C., Wang, Y., Nguyen, H.N.: Random forest classifier combined with feature selection for breast cancer diagnosis and prognostic. *J. Biomed. Sci. Eng.* **6**(05), 551 (2013)
20. Nilashi, M., Ahmadi, H., Shahmoradi, L., Ibrahim, O., Akbari, E.: A predictive method for hepatitis disease diagnosis using ensembles of neuro-fuzzy technique. *J. Infect. Public Health* **12**, 13 (2018)
21. Park, I.H., et al.: Disease-specific induced pluripotent stem cells. *Cell* **134**(5), 877–886 (2008)
22. Polivka, J., Kralickova, M., Kaiser, C., Kuhn, W., Golubnitschaja, O.: Mystery of the brain metastatic disease in breast cancer patients: improved patient stratification, disease prediction and targeted prevention on the horizon? *EPMA J.* **8**(2), 119–127 (2017)
23. Ruder, S.: An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098* (2017)
24. Shi, Z., Zuo, W., Chen, W., Yue, L., Han, J., Feng, L.: User relation prediction based on matrix factorization and hybrid particle swarm optimization. In: *Proceedings of the 26th International Conference on World Wide Web Companion*, pp. 1335–1341. International World Wide Web Conferences Steering Committee (2017)
25. Sicherer, S.H., Sampson, H.A.: Food allergy: a review and update on epidemiology, pathogenesis, diagnosis, prevention, and management. *J. Allergy Clin. Immunol.* **141**(1), 41–58 (2018)
26. Song, H., Rajan, D., Thiagarajan, J.J., Spanias, A.: Attend and diagnose: clinical time series analysis using attention models. *arXiv preprint arXiv:1711.03905* (2017)

27. Subasi, A.: Classification of EMG signals using PSO optimized SVM for diagnosis of neuromuscular disorders. *Comput. Biol. Med.* **43**(5), 576–586 (2013)
28. Tangri, N., et al.: A predictive model for progression of chronic kidney disease to kidney failure. *JAMA* **305**(15), 1553–1559 (2011)
29. Trask, A., Gilmore, D., Russell, M.: Modeling order in neural word embeddings at scale. arXiv preprint [arXiv:1506.02338](https://arxiv.org/abs/1506.02338) (2015)
30. Vaswani, A., et al.: Attention is all you need. In: *Advances in Neural Information Processing Systems*, pp. 5998–6008 (2017)
31. Zhang, D., Shen, D., Initiative, A.D.N., et al.: Multi-modal multi-task learning for joint prediction of multiple regression and classification variables in Alzheimer’s disease. *NeuroImage* **59**(2), 895–907 (2012)



# Learning $k$ -Occurrence Regular Expressions with Interleaving

Yeting Li<sup>1,2</sup>, Xiaolan Zhang<sup>1,2</sup>, Jialun Cao<sup>1,2</sup>, Haiming Chen<sup>1(✉)</sup>,  
and Chong Gao<sup>1,2</sup>

<sup>1</sup> State Key Laboratory of Computer Science, Institute of Software,  
Chinese Academy of Sciences, Beijing 100190, China

{liyt,zhangxl,caojl,cm,gaochong}@ios.ac.cn

<sup>2</sup> University of Chinese Academy of Sciences, Beijing, China

**Abstract.** Since lacking valid schemas is a critical problem for XML and present research on interleaving for XML is also quite insufficient, in this paper we focus on the inference of XML schemas with interleaving. Previous researches have shown that the essential task in schema learning is inferring regular expressions from a set of given samples. Presently, the most powerful model to learn XML schemas is the  $k$ -occurrence regular expressions ( $k$ -OREs for short). However, there have been no algorithms that can learn  $k$ -OREs with interleaving. Therefore, we propose an entire framework which can support both  $k$ -OREs and interleaving. To the best of our knowledge, our work is the first to address these two inference problems at the same time. We first defined a new subclass of regular expressions named  $k$ -OIREs, and developed an inference algorithm *i*KOIRE to learn  $k$ -OIRE based on genetic algorithm and *maximum independent set* (MIS). We further conducted a series of experiments on large-scale real datasets, and evaluated the effectiveness of our work compared with both ongoing learning algorithms in academia and industrial tools in real world. The results reveal the high practicability and outstanding performance of our work, and indicate its promising prospects in application.

**Keywords:** Regular expression · Language learning · Interleaving · XML schema

## 1 Introduction

Extensible Markup Language (XML), as a standard format for data representation and exchange, is ubiquitous in various applications on websites, and XML schema specifies how to formally describe the elements in a XML document. Though the presence of a schema for XML documents has many applications,

---

Work supported by the National Natural Science Foundation of China under Grant Nos. 61872339 and 61472405.

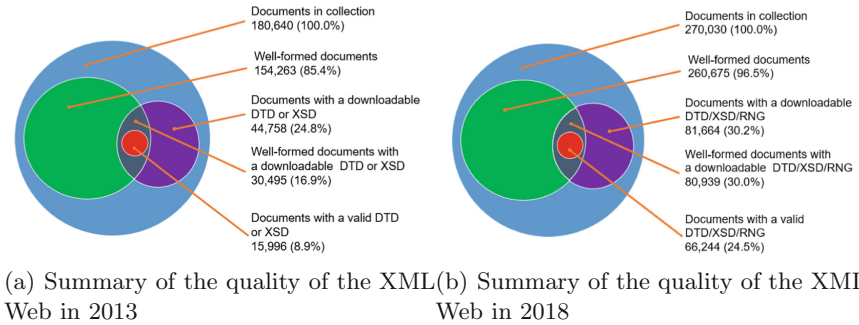
© Springer Nature Switzerland AG 2019

G. Li et al. (Eds.): DASFAA 2019, LNCS 11447, pp. 70–85, 2019.

[https://doi.org/10.1007/978-3-030-18579-4\\_5](https://doi.org/10.1007/978-3-030-18579-4_5)

such as data processing, automatic data integration and static analysis of transformations [1, 7, 21, 26–28, 32], the research on XML schemas especially for the unordered concatenation operator *interleaving* (shuffle) is relatively insufficient due to its difficulty. Therefore, in this paper, we focus on the inference of XML schemas with interleaving.

The practical situation of XML schemas, especially the valid ones, is unsatisfying. According to Grijzenhout *et al.* survey in 2013 [17], XML documents with corresponding DTD/XSD definitions on the Web account for 24.8%, with only 8.9% for valid ones. After five years, according to the latest investigation we conducted, the figures only witnessed a slight increase. The investigation was carried out on a much larger scale, and the source of XML files are more extensive. It is clear that the problem of lacking XML schemas still remains. Thus, schema inference is in desperate need to improve this situation. Besides, it is also useful in situations where a schema is already available, such as in schema cleaning and dealing with noise [3].



**Fig. 1. Summary of the quality of the XML Web.** DTD, XSD and Relax NG (abbrev. RNG) are popular schema languages for XML.

To address schema inference, the essential task is to infer regular expressions from a set of given samples [4, 5, 15]. In DTD language, the learned regular expressions (REs) can be used directly as part of the schema, while in XSD and RNG, they are the crucial components of schemas. However, a seminal result by Gold [16] shows that the class of REs could not be identifiable from positive examples only. Consequently, researchers have turned to study subclasses of REs and their corresponding inference algorithms. The well-studied subclasses of regular expressions include *single occurrence regular expressions* (SOREs), *chain regular expressions* (CHAREs),  *$k$ -occurrence regular expressions* ( $k$ -OREs) and so on. Among the learning algorithms, most of them can only deal with single occurrence regular expressions (i.e., expressions in which each symbol occurs at most once), e.g., [3, 4, 11, 13]. Later, a more powerful model,  *$k$ -occurrence regular expressions* ( $k$ -OREs) was proposed [2], which allows to learn regular expressions where each alphabet symbol occurs at most  $k$  times ( $k$  is usually small).

The presence of  $k$ -OREs is reasonable and sufficient in real world, since the study carried out by Li *et al.* revealed that when  $k$  is set to 7, almost 100% regular expressions will be covered in DTDs, XSDs and RNGs [22]. However,  $k$ -OREs do not support interleaving, while this operator is a contributing component of RNG. And in fact, in the literatures, there have been only very few work that support interleaving or  $k$ -occurrence [2, 6, 23, 25, 33, 39]:

- **$k$ -OREs without interleaving.** The inference of  $k$ -OREs is far more complicated than the single occurrence ones, so the work supporting interleaving is relatively limited.
- **Interleaving without  $k$ -OREs.** With interleaving, even for single occurrence regular expressions, the inference problem is already intractable, so most existing work used approximate algorithms to address this problem.

Despite the challenge, we propose the first framework which supports  $k$ -OREs and interleaving at the same time. Based on genetic algorithm and *maximum independent set* (MIS), we address this problem successfully. Firstly, we introduce the definition of a new subclass of regular expressions with interleaving, ( $k$ -OIREs). Then, we develop the corresponding learning algorithm, *i*KOIRE, to carry out  $k$ -OIREs inference automatically. To be more specific, we permit interleaving of the form  $c_1 \& c_2 \& \dots \& c_q$  where  $\&$  stands for interleaving and  $c_i$  represents particular Extended String (see Sect. 2 for their definitions). For example,  $r = (a|b^+)c^*((a^*b^?)\&(ab^*(c|d^+)^?))^?$  is a  $k$ -OIRE. The massive experimental results demonstrate the practicality of the proposed subclass as well as the outstanding performance of our work.

The main contributions of the paper are as follows:

- We develop a framework which can support both  $k$ -OREs and interleaving. To the best of our knowledge, our work is the first one to address these two inference problems at the same time. We hope our work may shed some lights on further research of their combination.
- We propose a new subclass of regular expressions with interleaving,  $k$ -OIREs. This subclass could cover the most Relax NG compared with the existing subclasses. Correspondingly, we also design an inference algorithm *i*KOIRE which can learn  $k$ -OIRE effectively based on genetic algorithm and MIS.
- We conduct a series of experiments, comparing the performance of both ongoing learning algorithms in academia and industrial tools in real-world. The results not only reveal the high practicability of  $k$ -OIRE and the effectiveness of *i*KOIRE, show the high preciseness and conciseness of our work, but also indicate its promising prospects.

The rest of this paper is organized as follows. Preliminaries are presented in Sect. 2. Section 3 provides the learning algorithm we used in this paper. Then a series of experiments is presented in Sect. 4. Finally we conclude this work in Sect. 5.

## 2 Preliminaries

Let  $\Sigma$  be a finite alphabet. The set of all words over  $\Sigma$  is denoted by  $\Sigma^*$ .  $\varepsilon$  denotes the empty word. A **regular expression with interleaving** over  $\Sigma$  is defined inductively as follows:  $\varepsilon$  or  $a \in \Sigma$  is a regular expression, for regular expressions  $r_1$  and  $r_2$ , the disjunction  $r_1|r_2$ , the concatenation  $r_1 \cdot r_2$ , the interleaving  $r_1 \& r_2$ , or the Kleene-Star  $r_1^*$  is also a regular expression. The size of a regular expression  $r$ , denoted by  $|r|$ , is the total number of symbols and operators occurred in  $r$ .  $r^?$  and  $r^+$  are abbreviations of  $r|\varepsilon$  and  $r \cdot r^*$ , respectively. The language  $L(r)$  of a regular expression  $r$  is defined as follows:  $L(\emptyset) = \emptyset$ ;  $L(\varepsilon) = \{\varepsilon\}$ ;  $L(a) = \{a\}$ ;  $L(r_1^*) = L(r_1)^*$ ;  $L(r_1 \cdot r_2) = L(r_1)L(r_2)$ ;  $L(r_1|r_2) = L(r_1) \cup L(r_2)$ ;  $L(r_1 \& r_2) = L(r_1) \& L(r_2)$ .

Let  $u = au'$  and  $v = bv'$  where  $a, b \in \Sigma$  and  $u', v' \in \Sigma^*$ . Then  $u \& \varepsilon = \varepsilon \& u = u$ .  $u \& v = \{a \cdot (u' \& v')\} \cup \{b \cdot (u \& v')\}$ . For example, strings accepted by  $(abc) \& (d)$  is the set  $\{abcd, dabc, adbc, abdc\}$ .

The marked form of regular expression  $r$  ( $\bar{r}$ ), is obtained by marking symbols in  $r$  with subscripts, such that each marked symbol occurs only once in  $\bar{r}$ . For an expression  $r = a(a|b)(ab)^*$ , its marked form can be  $a_1(a_2|b_1)(a_3b_2)^*$ . The same notation will also be used for dropping subscripts from the marked symbols:  $\bar{\bar{r}} = r$ . We extend this notation for words and sets of symbols in the obvious way.

The new subclass of regular expressions proposed is called  $k$ -occurrence regular expressions with interleaving ( $k$ -OIREs). Before introducing its definition, we first give a concept used in  $k$ -OIREs: *particular Extended String* ( $pES$ ).

**Definition 1. Particular Extended String ( $pES$ ).** Let  $\Sigma$  be a finite alphabet. A  $pES$  is a finite sequence  $f_1 f_2 \cdots f_{n_1}$ .  $f_i$  can be of two forms. One is of the form  $a^b$  where  $a \in \Sigma$ ,  $b \in \{1, ?, +, *\}$ . The other is  $(e_1|e_2|\cdots|e_{m_1})^t$  where  $t \in \{1, ?\}$  and  $e_j$  is of the form of  $a^b$ . For example,  $a^*b^?c^+d$  and  $a^*b(c|d^+)^?$  are both  $pES$ s.

**Definition 2.  $k$ -Occurrence regular expressions with interleaving ( $k$ -OIREs).** Let  $\Sigma$  be a finite alphabet.  $k$ -OIREs is a class of regular expressions over  $\Sigma$ , in which each terminal symbol can occur at most  $k$  times. It consists of a finite sequence of two kinds of factors. One kind is of the form  $(b_1|b_2|\cdots|b_m)^t$  where  $t \in \{1, ?\}$ ,  $m \geq 1$  and  $b$  is  $a$  or  $a^+$ ,  $a \in \Sigma$ . The other kind is of the form  $(d_1|d_2|\cdots|d_p)^t$  where  $t \in \{1, ?\}$ ,  $p \geq 1$  and  $d_i$  is of the form  $c_1 \& c_2 \& \cdots \& c_q$  where  $c_i$  is a  $pES$ ,  $q \geq 2$ . For example,  $r = (a|b^+)^* ((a^*b^?) \& (ab^*(c|d^+)^?))^?$  is a  $k$ -OIRE.

**Definition 3 [2].** A  $k$ -OA is a node-labeled graph  $G = (V, R, lab)$  where:

- $V$  is a finite set of nodes (also called states) with a distinguished source  $src$  and sink  $snk$ .
- $R$  is the edge relation:  $src$  has only outgoing edges;  $snk$  has only incoming edges; every  $v \in V \setminus \{src, snk\}$  is reachable by a path from  $src$  to  $snk$ .
- $lab$  is the labeling function with  $V \setminus \{src, snk\} \rightarrow \Sigma$ .
- there are at most  $k$  states with the same symbol in  $\Sigma$ .

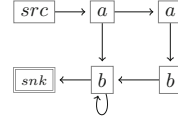
When  $k = 1$ , it is a 1-OA, also called the *single occurrence automaton* (SOA).

A **marked**  $k$ -OA  $\mathcal{A}$  is a  $k$ -OA in which each node is marked with a subscript such that each node label is unique in  $\mathcal{A}$ . Obviously a marked  $k$ -OA is an SOA.

A word  $a_1 \cdots a_n$  is accepted by automaton  $\mathcal{A}$  if there exists a path  $srcv_1 \cdots v_n snk$  in  $G$  such that  $a_i = lab(v_i)$  for  $1 \leq i \leq n$ . We denote the set of all words accepted by  $\mathcal{A}$  as  $L(\mathcal{A})$ . We use  $out_\sigma(v)$  to denote  $\{v_1 | (v, v_1) \in R \text{ and } \sigma = lab(v_1)\}$ , i.e., the set of states of all direct successors of a state  $v$  in  $\mathcal{A}$ .

**Definition 4** A  $k$ -OA is *deterministic*, if for any  $v \in V \setminus \{src, snk\}$  and  $\sigma \in \Sigma$ ,  $out_\sigma(v)$  contains at most one state.

The corresponding 2-OA for  $r = a(ab)^?b^+$  is shown in Fig. 2. It is deterministic clearly.



**Fig. 2.** A 2-OA for  $r = a(ab)^?b^+$

### 3 The Learning Algorithm

Given a finite set  $S$  of positive samples, we aim to learn a  $k$ -OIRE  $r$  with some fixed value of  $k$  such that  $S \subseteq L(r)$ . The learning algorithm (*i*KOIRE) is mainly based on genetic algorithm and maximum independent set (MIS). We show the major technical details of our algorithm in this section. The main executing processes are illustrated as follows.

1. For  $k \in [1, k_{max}]$ , generate a deterministic  $k$ -OA  $\mathcal{A}$  from  $S$  by Algorithm 2 GenKOA based on *genetic algorithm*.
2. We label each node in  $\mathcal{A}$  with a subscript to a *marked*  $k$ -OA  $\overline{\mathcal{A}}$ . Then, using  $\overline{\mathcal{A}}$ , we get a marked sample set  $\overline{S}$ . This process is executed using functions  $\text{MarkKOA}()$  and  $\text{MarkSample}()$ .
3. Based on  $\overline{\mathcal{A}}$  and  $\overline{S}$ , we convert  $\overline{\mathcal{A}}$  to a  $k$ -OIRE  $r$ . All these are implemented by Algorithm 4 GenKOIRE.

#### 3.1 The Main Algorithm

The pseudocode of the main algorithm is presented in Algorithm 1.

We have made many attempts with various  $k \in [1, k_{max}]$ , where  $k_{max}$  is the maximal number of occurrences of alphabets in  $S$ . Notably, after the analysis of real-world data, there are about 99.9% of practical REs in which the symbol can occur at most 7 times [22]. Therefore, we set  $k_{max} = 8$  in this paper. Different results will be learnt due to different values of  $k$ . The best one will be chosen considering two measures of regular expressions: *preciseness* and *conciseness*. Function  $\text{bestRE}()$  is to select an optimum result with certain value of  $k$ , according to two measures: **(a)** *Language Size* [4] and **(b)** *one part of the minimum description length (MDL)* [34].

*Language Size*, denoted by  $|\mathcal{L}(r)|$ , is defined as:  $|\mathcal{L}(r)| = \sum_{\ell=1}^{\ell_{max}} |L^\ell(r)|$ , where  $|L^\ell(r)|$  is the size of subset containing words with length  $\ell$  in  $L(r)$ . Generally,  $L(r)$  is an infinite language with infinitely large value of  $\ell$ , it is of course impossible to take all words into account. Hence, we only consider the word length  $\ell$  up to

a maximum value:  $\ell_{max} = 2m + 1$  where  $m$  is the length of  $r$  excluding  $\varepsilon$ ,  $\emptyset$  and regular expression operators. *Language Size* ( $|\mathcal{L}(r)|$ ) can well measure the preciseness of a regular expression. Smaller the value of  $|\mathcal{L}(r)|$  is, more precise the regular expression will be.

---

**Algorithm 1.** *i*KOIRE
 

---

**Input:** a sample  $S$   
**Output:** a  $k$ -ORE  $r$  with  $S \subseteq L(r)$

```

1 initialize candidate set  $C \leftarrow \emptyset$ 
2 for  $k = 1$  to  $k_{max}$  do
3   for  $n = 1$  to  $N$  do
4      $\mathcal{A} \leftarrow \text{GenKOA}(S, k)$ 
5      $\overline{\mathcal{A}} \leftarrow \text{MarkKOA}(\mathcal{A})$ 
6      $\overline{S} \leftarrow \text{MarkSample}(\overline{\mathcal{A}}, S)$ 
7      $r \leftarrow \text{GenKOIRE}(\overline{\mathcal{A}}, \overline{S})$ 
8     add  $r$  to  $C$ 
9 return  $r \leftarrow \text{bestRE}(C)$ 

```

---

Part of *MDL* illustrates the length of an expression  $r$  in bits, defined as:  $\mathcal{L}en(r) = |r| * \lceil \log_2(|\Sigma| + |\mathcal{M}|) \rceil$ , where  $|\Sigma|$  is the size of the alphabet and  $\mathcal{M}$  is the set of  $\{|\cdot, \cdot, \&, \cdot, *, +, (\cdot)\}$ . It reflects the conciseness of  $r$ . Similarly, smaller the value of  $\mathcal{L}en(r)$  is, more concise the regular expression will be.

Considering the above two indicators,  $\text{bestRE}()$  selects an optimum expression from the candidate set  $C$  with both the minimum values of  $|\mathcal{L}(r)|$  and  $\mathcal{L}en(r)$ .

Generally, optimization algorithms have to tackle the problem of local optimization, with no exception for the genetic algorithm. In order to decrease the probability of generating a local optimized result, we run the algorithms  $N$  times consequently. In the experiments we set  $N = 10$ .

### 3.2 Generate a Deterministic $k$ -OA from Samples

In this section, we introduce how the Algorithm 2 GenKOA learns a deterministic  $k$ -OA from given samples based on genetic algorithm.

**The Genetic Algorithm.** Here, we aim to find an optimum deterministic  $k$ -OA which can accept all strings in  $S$ , based on measures proposed above. In the genetic algorithm, each individual in the population will evolve better through *crossover* and *mutation* operations. After several generations, more adaptable individuals will appear. In our paper, each individual in the population is represented by a binary string. The process of the algorithm is an optimization of binary strings. And it can be divided into three steps.

**Initialization.** We first randomly initialize a population consisting of a number of binary strings. The length of the string is  $(k * |\Sigma| + 1)^2$ . Each string can be decoded into a  $k$ -OA easily, which is a possible solution with  $(k * |\Sigma| + 2)$  states and random edge relations. Take 2-OA for an example to explain the corresponding relation between the automaton and the binary string.



---

**Algorithm 2.** GenKOA

---

**Input:** a sample  $S$ , a  $k$  value  
**Output:** a deterministic  $k$ -OA  $\mathcal{A}$  with  $S \subseteq L(\mathcal{A})$

- 1  $P \leftarrow \text{init}(k), C \leftarrow \emptyset$
- 2 **for**  $g=1$  to  $g_{max}$  **do**
- 3      $koas \leftarrow \text{decode}(P)$
- 4      $values \leftarrow \text{calcFitness}(koas, S)$
- 5      $parents \leftarrow \text{select}(P, values)$
- 6      $P \leftarrow \text{crossover}(parents)$
- 7      $P \leftarrow \text{mutate}(P)$
- 8      $\mathcal{A} \leftarrow \text{bestFA}(\text{decode}(P), S)$
- 9      $\mathcal{A} \leftarrow \text{DISAMBIGUATE}(\mathcal{A}, S)$
- 10    add  $\text{SIMPLIFY}(\mathcal{A}, S)$  to  $C$
- 11 **return**  $\mathcal{A} \leftarrow \text{bestFA}(C)$

---



---

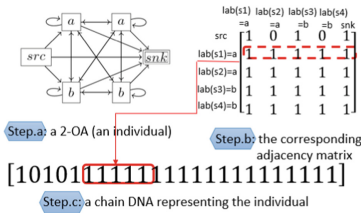
**Algorithm 3.** DISAMBIGUATE

---

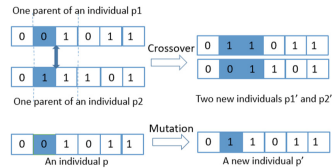
**Input:** a sample  $S$ , a  $k$ -OA  $\mathcal{A}$   
**Output:** a deterministic  $k$ -OA  $\mathcal{A}$

- 1 initialize queue  $Q$  to  $src$
- 2 initialize marked states set  $B \leftarrow \emptyset$
- 3 **while**  $|Q| \neq 0$  **do**
- 4      $s \leftarrow \text{first}(Q)$
- 5     **while**  $\exists \sigma \in \Sigma, |\text{out}_\sigma(s)| > 1$  **do**
- 6          $C \leftarrow \emptyset$
- 7         **for**  $t \in \text{out}_\sigma(s)$  **do**
- 8              $\mathcal{A}' \leftarrow \mathcal{A}$
- 9             **for**  $\forall t' \in \text{out}_\sigma(s) \setminus \{t\}$  **do**
- 10                 delete edge  $(s, t')$  from  $\mathcal{A}'$
- 11             add  $\mathcal{A}'$  to  $C$
- 12          $\mathcal{A} \leftarrow \text{bestFA}(C, S)$
- 13     add  $s$  to  $B$  and pop  $s$  from  $Q$
- 14     enqueue  $\forall \beta \in \text{out}(s) \setminus B$  to  $Q$
- 15 **return**  $\mathcal{A}$

---



**Fig. 3.** The process of encoding an individual



**Fig. 4.** Examples for crossover and mutation operations

Given a 2-OA in Fig. 3, we first compute its adjacency matrix  $W$ .  $W_{ij} = 1$  if there is an edge from node of row  $i$  to node of column  $j$ ,  $W_{ij} = 0$  otherwise. Then its corresponding binary string is obtained by concatenating each row numbers (with all columns) together. Similarly, we can decode a binary string to a  $k$ -OA in an inverse process.

P1: 1000010101100010001001010101010101010101010101010101010010000000110111010101  
 P2: 101000101011000001010010100010101011010101010011010000100101100011000010000  
 P3: 01100010001000100010010011010101010100010000010000010100001101010010010101111  
 P4: 1010010100110101011001100101010100011101000101010101000001101010010010001010100  
 P5: 1001101001010100000010100110010011011010000100001010001001010101010000011000111  
 P6: 010110100100110101010100001101000110001000110101010010101001001101001010011100

Fig. 5. Initialize the population P.

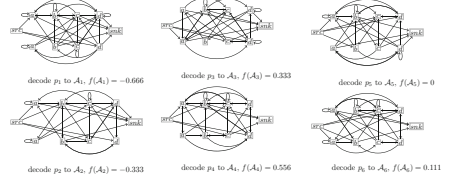


Fig. 6. Decode P to 2-OAs and calculate fitness.

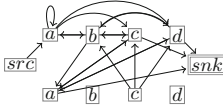


Fig. 7. bestFA

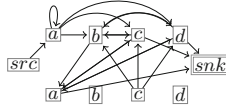


Fig. 8. DISAMBIGUATE

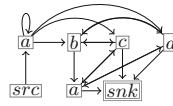


Fig. 9. SIMPLIFY

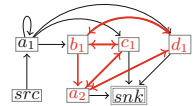


Fig. 10. Marked KOA

**Selection.** In each generation of the population, we select excellent individuals to breed new generations. We measure individuals by a *fitness* function used for finding preferential solutions. In our algorithm, the function will choose individuals which are representative enough to describe the class of languages of samples. Here the fitness function  $f(\mathcal{A})$  for an automaton  $\mathcal{A}$  is usually defined as:  $f(\mathcal{A}) = \frac{|T_P| - |F_N|}{|S|}$ , where  $T_P = \{w \in S \mid w \in L(\mathcal{A})\}$  (means true positive),  $F_N = \{w \in S \mid w \notin L(\mathcal{A})\}$  (means false negative),  $|S|$  is the size of the samples, respectively. To be mentioned, the computations of  $T_P$  and  $F_N$  involve the membership problem of checking whether a word  $w \in L(\mathcal{A})$  or not. We use the efficient algorithm in [18] to solve this problem. The  $f(\mathcal{A})$  guarantees that the selected individuals can accept strings in  $S$  as many as possible. For an automaton  $\mathcal{A}$ , it will be chosen with much higher possibility if  $f(\mathcal{A})$  is larger. The function  $\text{bestFA}()$  has the same effect as *select*. The only difference between them is that *select* returns a pair of *parents* from selected individuals based on the fitness function, while  $\text{bestFA}()$  only returns the best automaton  $\mathcal{A}$  (with the largest fitness value, i.e., 1, satisfying  $S \subseteq L(\mathcal{A})$ ).

**Crossover and Mutation.** They are two popular ways to evolve into the next generation of one population. For *crossover*, we first choose a pair of parents from the individuals selected above. Then determine the *starting* and *terminal* positions of each binary string. At last, we exchange the values between the two positions of two binary strings. For *mutation*, we can randomly choose a position in the binary string and change its value to the complementary number (i.e., 1 to 0). Figure 4 is an example.

DISAMBIGUATE can convert a  $k$ -OA into a deterministic  $k$ -OA. For each state  $s$  and symbol  $\sigma \in \Sigma$  such that  $|\text{out}_\sigma(s)| > 1$ , we delete edges to keep  $|\text{out}_\sigma(s)| = 1$ , which is guided by the function  $\text{bestFA}()$ . SIMPLIFY is designed to delete useless edges and states according to samples.

Here is an example. Suppose  $S = \{aacba, adba, abca, abda, abac, aabad\}$ ,  $\Sigma = \{a, b, b, d\}$ . We show the process of GenKOA to learn a deterministic 2-OA from  $S$ . In the first step, we randomly initialize the first generation population  $P$  (Fig. 5), consisting of 6 binary strings with length of  $(k * |\Sigma| + 1)^2 = 81$ . Each string is decoded into a 2-OA with 10 states. They are shown in Fig. 6, together with their fitness values, calculated by  $f(\mathcal{A}) = \frac{|T_P| - |F_N|}{|S|}$ . After 50 generations of *mutation* and *crossover*, we get the best 2-OA (Fig. 7). Using Algorithm 3 DISAMBIGUATE, we turn the 2-OA into a deterministic automaton (Fig. 8). After removing useless states and edges by function SIMPLIFY(), the deterministic automaton is simplified, named as  $\mathcal{A}$  (Fig. 9).

### 3.3 Mark the Deterministic $k$ -OA and Samples

In this section, we use  $\mathcal{A}$  obtained in Sect. 4.2 to mark each string in  $S$ , which is an important step for following steps. Before that, we first use MarkKOA() to get a marked  $k$ -OA  $\bar{\mathcal{A}}$  with subscripts for nodes in  $\mathcal{A}$  such that node labels in  $\bar{\mathcal{A}}$  are pairwise distinct. Then for each string  $s \in S$ , we can find a unique accepting path from  $\text{src}$  to  $\text{snk}$  in  $\mathcal{A}$ , denoted by  $\text{src} \cdot v_1 v_2 \cdots v_n \cdot \text{snk}$ . Obviously, this path can be uniquely mapped to a *marked* form in  $\bar{\mathcal{A}}$ :  $\text{src} \cdot v_{1k_1} v_{2k_2} \cdots v_{nk_n} \cdot \text{snk}$ , where  $v_i = \bar{v}_{ik_i}$ . After this operation, we obtain a marked strings set  $\bar{S}$ .

Following the example in Sect. 3.2, the marked  $k$ -OA of  $\mathcal{A}$  is shown in Fig. 10. And the marked strings set is  $\bar{S} = \{a_1 a_1 c_1 b_1 a_2, a_1 d_1 b_1 a_2, a_1 b_1 c_1 a_2, a_1 b_1 d_1 a_2, a_1 b_1 a_2 c_1, a_1 a_1 b_1 a_2 d_1\}$ .

### 3.4 Convert the Marked $k$ -OA into a $k$ -OIRE

Based on the marked  $k$ -OA  $\bar{\mathcal{A}}$  and marked strings set  $\bar{S}$ , we will introduce the algorithm GenKOIRE to convert  $\bar{\mathcal{A}}$  to a  $k$ -OIRE for a fixed  $k$  ( $k \in [1, k_{max}]$ ). The input is  $\bar{\mathcal{A}}$  and  $\bar{S}$ . The output is a  $k$ -OIRE  $r$ .

Following the example above, we first remove the self-loop in Fig. 10 and get a new SOA (Fig. 11). The maximum strongly connected component in Fig. 11 is  $\mathcal{C} = \{a_2, b_1, c_1, d_1\}$ . Using  $\mathcal{C}$ , we get a new set  $S' = \text{Filter}(\mathcal{C}, \bar{S}) = \{c_1 b_1 a_2, d_1 b_1 a_2, b_1 c_1 a_2, b_1 d_1 a_2, b_1 a_2 c_1, b_1 a_2 d_1\}$  [25]. Using  $S'$ , we compute the *constraint\_tr* [33] (Fig. 12) and construct its undirected graph  $G$  of *constraint\_tr* (Fig. 13). There are two *maximum independent sets* ( $\{a_2, b_1\}, \{c_1, d_1\}$ ) in  $G$ . String sets  $S''_1$  and  $S''_2$  are obtained from  $S'$  by Filter() using  $\{a_2, b_1\}, \{c_1, d_1\}$ , respectively. Two SOAs (Figs. 14 and 15) are constructed from  $S''_1$  and  $S''_2$  using *2T-INF* [14]. Each SOA is converted into a SORE by algorithm Soa2Sore [13]. A new RE is obtained by combining all SOREs with interleaving operator. The *maximum strongly connected component*, together with all relative edges, is replaced by this new RE and the marked  $k$ -OA  $\bar{\mathcal{A}}$  is updated shown in Fig. 16.

Finally, we compute level number for each node in Fig. 16 and find the skip levels [13]. Nodes of each level number are turned into one or more chain factors. If there are more than one non-letter nodes (node with more than one terminal symbols) with level number, or if the level number is a skip level, then ‘?’ is appended to every chain factor on that level. At last, we drop all subscripts and obtain the final 2-OIRE  $a^+(ba\&(c|d))$ .

---

**Algorithm 4.** GenKOIRE
 

---

**Input:** a marked  $k$ -OA  $\bar{\mathcal{A}}$ , a marked sample  $\bar{S}$   
**Output:** a  $k$ -OIRE  $r$

```

1  $\bar{\mathcal{A}} \leftarrow \bar{\mathcal{A}}.\text{RemoveSelfLoop}()$ 
2 while  $\bar{\mathcal{A}}$  has a cycle do
3    $\left| \text{Let } U \text{ be a strongly connected looped component of } \bar{\mathcal{A}} \right.$ 
4    $\left| \zeta \leftarrow \text{MERGE}(U, \bar{S}); \bar{\mathcal{A}} \leftarrow \bar{\mathcal{A}}.\text{contract}(U, \zeta) \right.$ 
5  $\bar{\mathcal{A}}.\text{constructLevelOrder}()$ 
6  $r \leftarrow \varepsilon$ 
7 for  $i = 1$  to (level number of  $\bar{\mathcal{A}}.\text{snk}$ )  $- 1$  do
8    $V_t \leftarrow$  all nodes with level number  $i$  and  $\&$ 
9    $V_s \leftarrow$  all nodes with level number  $i$  and no  $\&$ 
10  if  $|V_t| \geq 1$  then
11    if  $|V_s| \geq 1$  or  $\bar{\mathcal{A}}.\text{isSkipLevel}(i)$  then
12       $r \leftarrow r \cdot \text{ALT}(V_t)?$ 
13    else  $r \leftarrow r \cdot \text{ALT}(V_t)$ 
14  if  $|V_s| \geq 1$  then
15    if  $|V_t| \geq 1$  or  $\bar{\mathcal{A}}.\text{isSkipLevel}(i)$  then
16       $r \leftarrow r \cdot \text{ALT}(V_s)?$ 
17    else  $r \leftarrow r \cdot \text{ALT}(V_s)$ 
18 return  $r \leftarrow \text{drop}(r)$ 
```

---



---

**Algorithm 5.** MERGE
 

---

**Input:** a set of nodes  $U$ , a sample  $S$   
**Output:** an expression  $\zeta$

```

1  $S' \leftarrow \text{Filter}(U, S)$ ;  $\text{constraint\_tr} \leftarrow \text{CS}(S')$ ;  $G \leftarrow \text{Graph}(\text{constraint\_tr})$ 
2  $\text{all\_mis} \leftarrow \emptyset$ 
3 while  $|G.\text{nodes}()| > 0$  do
4    $\left| W \leftarrow \text{clique\_removal}(G)$ ;  $G \leftarrow G \setminus W$ ;  $\text{all\_mis.append}(G) \right.$ 
5  $\text{SubRE} \leftarrow \emptyset$ 
6 foreach  $\text{mis}$  in  $\text{all\_mis}$  do
7    $S'' \leftarrow \text{Filter}(\text{mis}, S)$ ;  $\mathcal{A}_1 \leftarrow \text{SOA}(S'')$ ;  $\text{sub\_re} \leftarrow \text{Soa2Sore}(\mathcal{A}_1)$ 
8   if  $\varepsilon \in S''$  then
9      $\left| \text{SubRE.append}(\text{sub\_re}?) \right.$ 
10  else  $\text{SubRE.append}(\text{sub\_re})$ 
11  $\zeta \leftarrow \text{combine}(\text{SubRE})$ 
12 return  $\zeta$ ;
```

---

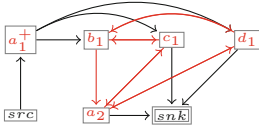


Fig. 11. Remove self-loop

<i>constraint_tr</i>	
$\langle a_2, c_1 \rangle$	$\langle c_1, a_2 \rangle$
$\langle a_2, d_1 \rangle$	$\langle d_1, a_2 \rangle$
$\langle b_1, c_1 \rangle$	$\langle c_1, b_1 \rangle$
$\langle b_1, d_1 \rangle$	$\langle d_1, b_1 \rangle$

Fig. 12. *constraint\_tr*

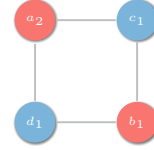


Fig. 13. Undirected graph

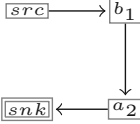


Fig. 14. SOA  $A_1$

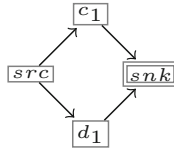


Fig. 15. SOA  $A_2$

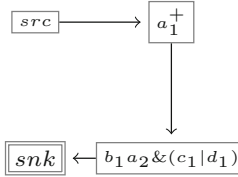


Fig. 16. New marked  $k$ -OA

## 4 Experiments

In this section, we conduct a series of experiments to analyze the practicability of our work, and compare our work with not only the learning algorithms from ongoing researches but also the industrial-level tools used in real world. In terms of preciseness and conciseness, our work has achieved satisfying results compared with existing methods, reaching higher preciseness with less description length. Specifically, indicators *Language Size* ( $|\mathcal{L}(r)|$ ) [2] and *datacost* (**DC**) [2] are used to measure preciseness, while length of regular expressions (**Len**) and *Nesting Depth* (**ND**) [24] for conciseness. Similar as the discussion of  $|\mathcal{L}(r)|$  and **Len** above, we have that larger the value of DC (ND) is, more precise (concise) the regular expression will be. The learning algorithms used for experiments are Soa2Sore [13] and Soa2Chare [13], GenEchare [11], *iDREGEX* [2], *learner*<sub>DME</sub><sup>+</sup> [8], conMiner [33], GenICHARE [39] and GenESIRE [25]. Among them, the first three adopt standard regulation expression, *iDREGEX* exploits  $k$ -occurrence, and the rest ones involve the interleaving. The industrial tools which are capable of supporting inference of XML schemas used in this section include XmlSchemaInference [30], Stylus Studio [35], FreeFormatter [12], devtilsonline [9], mherman [29], EditiX [10], IntelliJ IDEA [20], Liquid Studio [36], Trang [37], InstanceToSchema [19], Oxygen XML [31] and XMLBlueprint [38]. Among these practical tools, only about half of them are supportive of Relax NG, and the supporting levels and methods vary.

For the massive comparative experiments, we conduct the experiments based on two kinds of datasets: small dataset (i.e., *mastersthesis*) and large dataset (i.e., *www*) of XML documents, which are both extracted from DBLP. DBLP is a data-centered database of information on major computer science journals and proceedings. We download the file of version *dblp-2015-03-02.xml.gz*<sup>1</sup>.

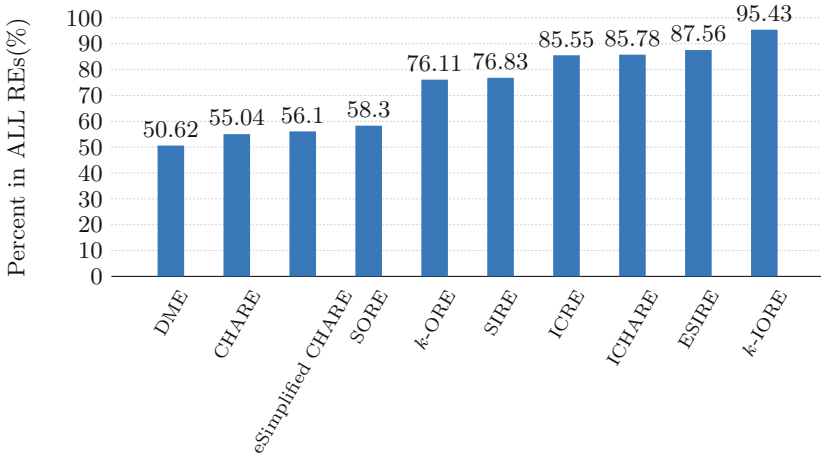
<sup>1</sup> <http://dblp.org/xml/release/dblp-2015-03-02.xml.gz>.

*mastersthesis* and *www* are two elements chosen from DBLP with 5 (small) and 2,000,226 (large) samples, respectively.

All our experiments are conducted on a machine with 16 cores Intel Xeon CPU E5620 @ 2.40 GHz with 12M Cache, 24G RAM, OS: Windows 10. The configurations are set as follow. We initialize the size of population  $P$  by randomly generating 300 DNAs, the maximum number of generation evolution  $g_{max}$  is set as 100. The parameters *crossover rate* and *mutation rate* are assigned as 0.8 and 0.003 according to the experience.

#### 4.1 Analysis on Practicality

Though interleaving is indispensable in data-centric applications, the lack of attention and research on it is still a concern. In Fig. 17, we visualized the coverage rates of REs covered by different subclasses on Relax NG. We can see that the initial subclass, DME, only covers 50.62%. Then the proportions show an upward trend, reaching more than 85.55% (ICRE, ICHARE, ESIRE). Compared with their coverage,  $k$ -OIRE covers 95.43% Relax NG, which is about 10% more than the second largest proportion. Therefore, the experimental result reveals the high practicality of  $k$ -OIRE, and its strong support for interleaving.



**Fig. 17. The proportion of subclasses on Relax NG.** The dataset used for this statistical experiment is acquired from [22], with 509,267 REs from 4,526 RNGs.

#### 4.2 Comparison with Learning Algorithms

To better illustrate the performance of our work, we first compare the inferred results of our work with that of existing learning algorithms. The experimental results are shown in Table 1.

We can see from Table 1(a) that for dataset *mastersthesis*, the first three algorithms (Soa2Sore, Soa2Chare and GenEchare) reach high conciseness at

enormous cost of  $|\mathcal{L}(r)|$ , from unaffordable  $1.0028 * 10^{64}$  to  $1.6383 * 10^4$ . Algorithms  $learner_{DME}^+$  and conMiner have highest conciseness, with 52 for  $\mathcal{L}en$ , yet their preciseness is not the highest among these algorithms. On the contrary, although with the highest precision, algorithm  $iDREGEX$  is the only one whose ND is 2, which lowers its conciseness. Finally, the last three algorithms including  $iKOIRE$  reach the performance at the same level, with highest preciseness and the equal magnitude of conciseness. Note that among algorithms with shortest  $|\mathcal{L}(r)|(5)$ ,  $iDREGEX$  exploits  $k$ -occurrence, while the last three ones adopt interleaving. From the table we can draw a conclusion that though both  $k$ -occurrence and interleaving could improve the preciseness, the former one sacrifices the conciseness to some degree. For the second dataset (Table 1(b)), the difference in performance is more obvious. Without supporting the usage of interleaving, the previous five algorithms have huge  $|\mathcal{L}(r)|$  and DC, from  $1.0028 * 10^{64}$  to  $9.9478 * 10^9$  and from 21804.719 to 4299.165, respectively. Among them, Soa2Chare has the shortest  $\mathcal{L}en$ , which is 120, while  $iDREGEX$  has the longest, which is 295. Soa2Sore has the deepest ND [24], with 3, followed by GenEchare and  $iDREGEX$ , with 2 nestings. On the other hand, the algorithms which support interleaving have smaller figures on average. Especially for the indicator  $|\mathcal{L}(r)|$ , the magnitudes are much smaller than that of the first group of methods. It is noteworthy that our work reaches almost the same conciseness with much less  $|\mathcal{L}(r)|(2.6595 * 10^8)$  and DC(4242.066), with about one hundredth of the second smallest  $|\mathcal{L}(r)|$  and half of the smallest DC.

It is clear from the above analysis, our work outperforms other state-of-the-art learning algorithms, achieving the highest preciseness and the equal level of conciseness. Furthermore, through the comparison, the performance of our method indicates that methods using interleaving present higher capability than others, and that the involvement of interleaving could contribute to both preciseness and conciseness.

**Table 1.** Results of inference using different learning algorithms

(a) mastersthesis (DBLP)					(b) www (DBLP)				
Algorithms	$ \mathcal{L}(r) $	DC	$\mathcal{L}en$	ND	Algorithms	$ \mathcal{L}(r) $	DC	$\mathcal{L}en$	ND
Original	$1.0028 * 10^{64}$	510.342	240	1	Original	$1.0028 * 10^{64}$	21804.719	240	1
Soa2Sore	$1.6383 * 10^4$	67.657	56	1	Soa2Sore	$1.3031 * 10^{12}$	7190.139	165	3
Soa2Chare	$1.6383 * 10^4$	67.657	56	1	Soa2Chare	$1.3553 * 10^{19}$	13696.752	120	1
GenEchare	$1.6383 * 10^4$	67.657	56	1	GenEchare	$1.3365 * 10^{19}$	13685.703	150	2
$iDREGEx$	5	80.500	80	2	$iDREGEx$	$9.9478 * 10^9$	4299.165	295	2
$learner_{DME}^+$	984	102.446	52	1	$learner_{DME}^+$	$1.4338 * 10^{15}$	11150.850	175	1
conMiner	13	72.886	52	1	conMiner	$4.1147 * 10^{13}$	10453.822	145	1
GenICHARE	5	65.072	60	1	GenICHARE	$1.4056 * 10^{13}$	9961.492	170	2
GenESIRE	5	65.072	60	1	GenESIRE	$4.3899 * 10^{11}$	8479.873	175	2
$iKOIRE$	5	65.072	60	1	$iKOIRE$	<b><math>2.6595 * 10^8</math></b>	<b>4242.066</b>	<b>265</b>	<b>1</b>

### 4.3 Comparison with Industrial Tools

Apart from ongoing research approaches, we also compare our work with industrial tools in real-world. The experiment results indicate the outstanding performance and promising application prospects of our work.

We can see from Table 2(a) that apart from InstanceToSchema ( $|\mathcal{L}(r)|$ : 984) and our work ( $|\mathcal{L}(r)|$ : 5), other tools cost extremely huge  $|\mathcal{L}(r)|$  (from  $1.6383 * 10^4$  to  $1.0028 * 10^{64}$ ) to reach the equal level of conciseness. Even for the tools such as Stylus Studio, mherman, IntelliJ IDEA and Trang with the third shortest  $|\mathcal{L}(r)|$ , the figures are more than 3000 times more than ours. Even compared with InstanceToSchema, our  $|\mathcal{L}(r)|(5)$  is still much shorter than theirs(984). While for conciseness, these tools are almost at the same level, except for XmlSchemaInference and EditiX, whose ND is 2. For the second dataset, the advantage of our work is more outstanding. While other tools have tremendous  $|\mathcal{L}(r)|$  and DC, our work only need  $2.6595 * 10^8$  for  $|\mathcal{L}(r)|$  and 4242.066 for DC. It means that no matter how complicated the regular expression is, our work can still achieve stable results with both high preciseness. Though the  $\mathcal{L}en$  of our work is the largest, the gap is relatively small, especially compared with the huge gap in  $|\mathcal{L}(r)|$ .

In conclusion, even compared with published tools in real world, our work still outweighs the performances in terms of preciseness to the large extent, and reaches the same level as the best conciseness. These results show the promising prospects of our work, as well as the high possibility of application in practical.

**Table 2.** Results of inference using different schemas inference tools

(a) mastersthesis (DBLP)					(b) www (DBLP)				
Tools	$ \mathcal{L}(r) $	DC	$\mathcal{L}en$	ND	Tools	$ \mathcal{L}(r) $	DC	$\mathcal{L}en$	ND
Original	$1.0028 * 10^{64}$	510.342	240	1	Original	$1.0028 * 10^{64}$	21804.719	240	1
XmlSchemaInference	$1.5673 * 10^{10}$	122.880	80	2	XmlSchemaInference	$1.1111 * 10^{21}$	15158.773	160	2
Stylus Studio	$1.6383 * 10^4$	67.657	56	1	Stylus Studio	$1.2047 * 10^{19}$	13606.698	125	1
FreeFormatter/devutilsonline	$1.5673 * 10^{10}$	122.880	56	1	FreeFormatter/devutilsonline	$1.1111 * 10^{21}$	15158.773	110	1
mherman	$1.6383 * 10^4$	67.657	56	1	mherman	$1.2047 * 10^{19}$	13606.698	125	1
EditiX	$1.5673 * 10^{10}$	122.880	80	2	EditiX	$1.1111 * 10^{21}$	15158.773	160	2
IntelliJ IDEA	$1.6383 * 10^4$	67.657	56	1	IntelliJ IDEA	$1.1859 * 10^{19}$	13696.31283	120	1
Liquid Studio	$1.5673 * 10^{10}$	122.880	56	1	Liquid Studio	$1.1111 * 10^{21}$	15158.773	120	2
Trang	$1.6383 * 10^4$	67.657	56	1	Trang	$1.2047 * 10^{19}$	13606.698	125	1
InstanceToSchema	984	102.446	52	1	InstanceToSchema	$1.5339 * 10^{18}$	13406.824	145	1
Oxygen XML/XMLBlueprint	$1.6383 * 10^4$	67.657	56	1	Oxygen XML/XMLBlueprint	$1.2047 * 10^{19}$	13606.698	125	1
<b>iKOIRE</b>	<b>5</b>	<b>65.072</b>	<b>60</b>	<b>1</b>	<b>iKOIRE</b>	<b><math>2.6595 * 10^8</math></b>	<b>4242.066</b>	<b>265</b>	<b>1</b>

## 5 Conclusions

In this paper, we proposed the first framework which can support both  $k$ -occurrence and interleaving at the same time. Starting from defining a new subclass of regular expressions with interleaving,  $k$ -OIREs, we designed an inference algorithm *iKOIRE* correspondingly. We further conducted a series of experiments to evaluate the performance of our work. By comparing with ongoing learning algorithms in academia and industrial tools in real world, our work presents the outstanding results, with the highest preciseness and conciseness. The experiment results reveal the effectiveness of our work in both research and practical scenario.



## References

1. Benedikt, M., Fan, W., Geerts, F.: XPath satisfiability in the presence of DTDs. *J. ACM* **55**(2), 8:1–8:79 (2008)
2. Bex, G.J., Gelade, W., Neven, F., Vansummeren, S.: Learning deterministic regular expressions for the inference of schemas from XML data. *TWEB* **4**(4), 14:1–14:32 (2010)
3. Bex, G.J., Neven, F., Schwentick, T., Tuyls, K.: Inference of concise DTDs from XML data. In: *Proceedings of the 32nd VLDB*, pp. 115–126 (2006)
4. Bex, G.J., Neven, F., Schwentick, T., Vansummeren, S.: Inference of concise regular expressions and DTDs. *ACM Trans. Database Syst.* **35**(2), 11:1–11:47 (2010)
5. Bex, G.J., Neven, F., Vansummeren, S.: Inferring XML schema definitions from XML data. In: *Proceedings of the 33rd VLDB*, pp. 998–1009 (2007)
6. Boneva, I., Ciucanu, R., Staworko, S.: Simple schemas for unordered XML. In: *Proceedings of the 16th WebDB*, pp. 13–18 (2013)
7. Che, D., Aberer, K., Özsu, M.T.: Query optimization in XML structured-document databases. *VLDB J.* **15**(3), 263–289 (2006)
8. Ciucanu, R., Staworko, S.: Learning schemas for unordered XML. In: *Proceedings of the 14th DBPL* (2013)
9. devutilsonline: Free XML to XSD Generator, March 2018. <https://devutilsonline.com/xsd-xml/generate-xsd-from-xml>
10. EditiX: Open Source XML Editor, March 2018. <https://www.editix.com/>
11. Feng, X.Q., Zheng, L.X., Chen, H.M.: Inference algorithm for a restricted class of regular expressions. *Comput. Sci.* **41**, 178–183 (2014)
12. freeformatter: XML Schema Generator, March 2018. <https://www.freeformatter.com/xsd-generator.html>
13. Freydenberger, D.D., Kötzing, T.: Fast learning of restricted regular expressions and DTDs. *Theor. Comput. Syst.* **57**(4), 1114–1158 (2015)
14. García, P., Vidal, E.: Inference of k-testable languages in the strict sense and application to syntactic pattern recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **12**(9), 920–925 (1990)
15. Garofalakis, M.N., Gionis, A., Rastogi, R., Seshadri, S., Shim, K.: XTRACT: learning document type descriptors from XML document collections. *Data Min. Knowl. Discov.* **7**(1), 23–56 (2003)
16. Gold, E.M.: Language identification in the limit. *Inf. Control* **10**(5), 447–474 (1967)
17. Grijzenhout, S., Marx, M.: The quality of the XML web. *J. Web Semant.* **19**, 59–68 (2013)
18. Hopcroft, J.E., Motwani, R., Ullman, J.D.: *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Boston (2001)
19. InstanceToSchema: RELAX NG Schema Generator, October 2003. <http://www.xmloperator.net/i2s/>
20. JetBrains: Capable and Ergonomic IDE for JVM, March 2018. <https://www.jetbrains.com/idea/>
21. Koch, C., Scherzinger, S., Schweikardt, N., Stegmaier, B.: Schema-based scheduling of event processors and buffer minimization for queries on structured data streams. In: *Proceedings of the 30th VLDB*, pp. 228–239 (2004)
22. Li, Y., Chu, X., Mou, X., Dong, C., Chen, H.: Practical study of deterministic regular expressions from large-scale XML and schema data. In: *Proceedings of the 22nd IDEAS*, pp. 45–53 (2018)

23. Li, Y., Mou, X., Chen, H.: Learning concise relax NG schemas supporting interleaving from XML documents. In: Proceedings of the 14th ADMA, pp. 303–317 (2018)
24. Li, Y., Zhang, X., Peng, F., Chen, H.: Practical study of subclasses of regular expressions in DTD and XML schema. In: Proceedings of the 18th APWeb, pp. 368–382 (2016)
25. Li, Y., Zhang, X., Xu, H., Mou, X., Chen, H.: Learning restricted regular expressions with interleaving from XML data. In: Proceedings of the 37th ER, pp. 586–593 (2018)
26. Manolescu, I., Florescu, D., Kossmann, D.: Answering XML queries on heterogeneous data sources. In: Proceedings of the 27th VLDB, pp. 241–250 (2001)
27. Martens, W., Neven, F.: Typechecking top-down uniform unranked tree transducers. In: Proceedings of the 9th ICDT, pp. 64–78 (2003)
28. Martens, W., Neven, F.: Frontiers of tractability for typechecking simple XML transformations. *J. Comput. Syst. Sci.* **73**(3), 362–390 (2007)
29. mherman: XML Schema Generator, March 2018. <http://xml.mherman.org/>
30. Microsoft: Xml Schema Inference - Developer Network, March 2018. <https://msdn.microsoft.com/en-us/library/system.xml.schema.xmlschemainference.aspx>
31. Oxygen: XML Editor, March 2018. <https://www.oxygenxml.com/>
32. Papakonstantinou, Y., Vianu, V.: DTD inference for views of XML data. In: Proceedings of the 19th PODS, pp. 35–46 (2000)
33. Peng, F., Chen, H.: Discovering restricted regular expressions with interleaving. In: Proceedings of the 17th APWeb, pp. 104–115 (2015)
34. Quinlan, J.R., Rivest, R.L.: Inferring decision trees using the minimum description length principle. *Inf. Comput.* **80**(3), 227–248 (1989)
35. StylusStudio: XML Integrated Development Environment (XML IDE), March 2018. <http://www.stylusstudio.com/>
36. liquid technologies: Graphical XML Editor, March 2018. <https://www.liquid-technologies.com/>
37. Trang: Multi-Format Schema Converter Based on RELAX NG, October 2008. <http://www.thaiopensource.com/relaxng/trang.html>
38. XMLBlueprint: XML Editor, March 2018. <https://www.xmlblueprint.com/>
39. Zhang, X., Li, Y., Cui, F., Dong, C., Chen, H.: Inference of a concise regular expression considering interleaving from XML documents. In: Proceedings of the 22nd PAKDD, pp. 389–401 (2018)



# Learning from User Social Relation for Document Sentiment Classification

Kangzhi Zhao<sup>(✉)</sup>, Yong Zhang, Yan Zhang, Chunxiao Xing, and Chao Li

RIIT, Beijing National Research Center for Information Science and Technology,  
Department of Computer Science and Technology, Institute of Internet Industry,  
Tsinghua University, Beijing 100084, China  
{zkz15,zhang-yan14}@mails.tsinghua.edu.cn,  
{zhangyong05,xingcx,li-chao}@tsinghua.edu.cn

**Abstract.** Sentiment analysis is a fundamental problem in the field of natural language processing. Existing methods incorporate both semantics of texts and user-level information into deep neural networks to perform sentiment classification of social media documents. However, they ignored the relations between users which can serve as a crucial evidence for classification. In this paper, we propose SRPNN, a deep neural network based model to take user social relations into consideration for sentiment classification. Our model is based on the observation that social relations between users with similar sentiment trends provide important signals for deciding the polarity of words and sentences in a document. To make use of such information, we develop a user trust network based random walk algorithm to capture the sequence of users that have similar sentiment orientation. We then propose a deep neural network model to jointly learn the text representation and user social interaction. Experimental results on two popular real-world datasets show that our model significantly outperforms state-of-the-art methods.

## 1 Introduction

With the popular social media such as microblog services and review sites, users can conveniently share their personal feelings and opinions on the Internet, and embed their characteristics and preferences into the subjective text [32]. Given a collection of documents, the task of sentiment classification is to infer the sentiment polarity or intensity of each document. With the rapid growth of social media data, sentiment classification has drawn much attention from research communities in recent years [23, 25, 33], which arises in many real world applications such as opinion mining, personalized recommendation and market analysis.

Early studies in this area mainly adopt feature-based method and construct classifiers to solve this problem. Pang and Lee [18] first adopted supervised learning method to build classifiers. Many studies [10, 12, 21] tried to integrate various types of features to enhance the effectiveness. Despite the plausible success of some shallow learning methods, feature engineering is labor-intensive.

Many models need domain-specific features, which make it difficult apply them to other datasets or applications.

Recently there have been some neural network based studies to extract features automatically so as to avoid the complicated feature engineering. Some previous studies focus on designing effective models for classification [2, 22], while others aim at learning better representations of the text [9, 28]. Recent studies [27, 33] further integrate features other than text representation, such as user personality and product information into the model to enhance the effectiveness. However, such studies also suffer from the data sparsity problem. For example, in the product review rating, one product or topic has only a few reviews from a user, which makes it difficult to develop an accurate predication of the sentiment.

To address this problem, we argue that the social relations between users can be adopted to augment the data so as to provide important signals for sentiment analysis. Our idea is based on two observations. Firstly, users have specific preferences on providing sentiment ratings. And users with similar sentiment orientation tend to have similar comments on one product or event. For example, if a user posts a tweet saying “Trump is the one who changes America”, it is difficult to judge his sentiment trend towards Trump. However, if we know that he has many followers who are against Trump, we can infer that it is very likely that he is against Trump, too. Secondly, a user with high authority (“opinion leader”) provides strong signals on the sentiment. For example, a user complains that his cellphone is easily overheating. He praises this brand of cellphones as “good at its warmth in winter”. It is difficult to identify this ironic negative comments from just the texts. However, if the user follows a tech leader who also blames about the overheating of his cellphone in studies, it is easier to obtain the user’s sentiment orientation to this cellphone by considering his interactions with the tech leader. Therefore, by constructing **user document** with social relations from above two aspects, the problem of data sparsity in original documents can be extensively alleviated.

In this paper, we propose **Social Relation Powered Neural Network (SRPNN)** model to utilize the user social relations for document-level sentiment classification. We first model user social relations as a user trust network and then propose a random walk algorithm to generate user documents from the network based on both user authority and sentiment similarity. We then propose a deep neural network model which exploits both the semantic representation of texts and user document to predict the sentiment orientation. To the best of our knowledge, this is the first work to jointly learn text representation and user social relations for sentiment classification.

We evaluate SRPNN on two popular datasets Twitter and Yelp and compare it with several state-of-the-art methods. Experimental results show that SRPNN outperforms various baseline methods including both feature-based approaches and deep learning based method. It also demonstrated the effectiveness of incorporating user social relations. The contributions of this paper are summarized as following:

- We propose a novel model SRPNN by leveraging the user social relations for document-level sentiment classification. To the best of our knowledge, this is the first work that combines the text representation and features of user social relation as the input of deep neural networks.
- We design a random walk based algorithm to generate high quality user document considering both user authority and sentiment similarity.
- We conduct extensive sets of experiments on two popular datasets. The experimental results demonstrate the effectiveness of our model.

## 2 Related Work

**Trust Learning in User Network.** Random Walk is an algorithm that generates a sequence of visited nodes by iteratively selecting a random neighbor of current node. It has been widely used in the applications like collaborate filtering [3] and personalized recommendation [14]. A large number of studies also adopt the idea of random walk [29,35]. DeepWalk [19] adopts the neural language model to learn the embedding of network and terminates the random walk sequence by setting a maximum step size. TidalTrust [4] utilizes a BFS algorithm to search the trust score between users in a network. TrustWalker [6] proposed a random walk based framework for recommendation problems.

**Sentiment Classification.** There is a long stream of studies for sentiment classification on documents. Pang and Lee [18] proposed a supervised learning framework for sentiment classification. Many studies design rich features to enhance the effectiveness, such as bag of opinion [21], product information [10] and sentiment lexicon [7]. Some studies [12] focused on integrating emotional signals into machine learning framework for sentiment analysis. Hu et al. [5] utilized matrix manipulations to address the noises in microblog texts and construct sentiment relations. Zhu et al. [38] focused on improving the efficiency of sentiment analysis in large scale of social networks.

Many studies adopted data-driven approaches to avoid handcrafted features [30,36]. Mikolov et al. [15] utilized the context information to train the word and phrase embedding. Le and Mikolov [9] introduced paragraph embedding. Socher et al. [22] and Dong et al. [2] proposed recursive deep neural networks for sentiment classification. Mishra et al. [16] adopted convolutional neural network, while Tang et al. [24] and Qian et al. [20] adopted recurrent neural network for sentiment analysis. Recently attention mechanism [17] is also widely used in multiple NLP tasks especially in sentiment classification [11,13,34].

**Personalized Sentiment Classification.** Personalized Sentiment Classification has become a popular topic recently. Tang et al. [28] incorporated sentiment information when learning the word embedding. Tang et al. [27] obtained richer feature for neural network by modeling the personality of users. They further integrated the product information and the aspect level information to help improve the effect of classification [25,26]. Chen et al. [1] adopted selectivity attention to model the relation between users and products. Song et al. [23] adopted user’s

following relation matrix to extend Latent Factor Model in microblog sentiment classification. Wu and Huang [33] constructed a personalized classifier to integrate user’s social network to sentiment classification. Zhao et al. [37] introduced a network embedding learning framework on heterogeneous microblog network. Wang et al. [31] incorporated user’s cross-lingual sentiment consistency with a multi-task learning framework to enrich the user post representation.

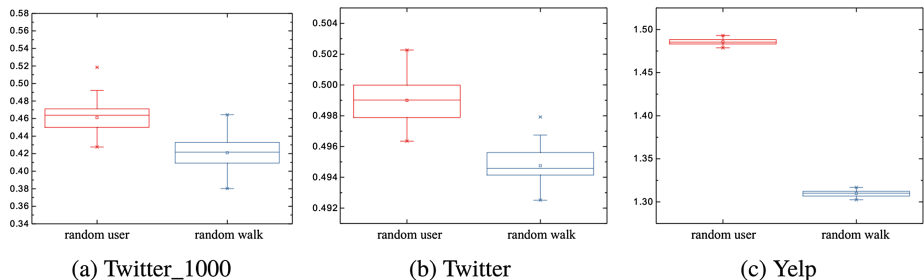


Fig. 1. User text-sentiment consistency for random walk

### 3 Constructing User Relation Sequence

In this section, we introduce the model to construct user relation sequences. We first justify the rationality to generate a series of user sequences using random walk and propose a user trust network to model the social similarity between users. We then take advantage of the network and devise a trust score as the metric to generate user relation sequence. Finally we propose a random walk based algorithm to effectively obtain the user relation sequences.

#### 3.1 User-Sentiment Consistency Verification

We have the observation that one user tends to produce documents with similar sentiment orientation. For example, a harsh user tends to evaluate the weaknesses of products, while an amiable user may focus on the advantages. Following this route, we find that users with similar sentiment orientation always tend to have similar comments on one product or event. We further argue that the sequence of user with user-sentiment consistency can be obtained using random walk algorithm. This can serve as the foundation of our model. Next we will validate this assumption.

To test the consistency assumptions of the random walk algorithm, for each user in Twitter and Yelp datasets we test for  $n = 50$  iterations. Given a user  $u_i$ , we randomly pick out one review rating  $r_j$ . If  $u_i$  has adjacent users, we pick another review rating  $r_j^{rw}$  from a user  $u_i^{rw}$  in the sequence of  $u_i$  in the similar way of executing random walk algorithm. We then randomly pick another

user  $u_j^{random}$  and his review rating  $r_j^{random}$ . By iteratively calculating the absolute difference  $(r_j, r_j^{rw})$  and  $(r_j, r_j^{random})$ , we can see the statistical discrepancy between random walk user and random user.

The test results of all three datasets are shown in Fig. 1. We can see that random walk users hold a lower difference of review rating than random users. Such results confirms the sentiment consistency of the user sequence generated by random walk algorithm.

### 3.2 User Trust Scoring

Although random walk algorithm can get a better result than selecting random user, we hope to further improve the sentiment consistency. Following the above assumption, we argue that *users with similar sentiment orientation tend to have similar comments on one product or event*. We call such users *trust users* and introduce a user trust network to model their relations. Figure 2 shows an example of the user trust network. The nodes are individual users, with the relationship “user  $u_i$  follows user  $u_j$ ” resulting in an edge directed from node  $u_i$  to node  $u_j$ . The out-degree of a node denotes the number of people a user follows. The weight  $\bar{r}_{u_i}$  on node  $u_i$  is its average rating. We utilize the degree of trust  $\mathcal{T}(u_i, u_j)$  on the adjacent users in the user trust network to denote the weight of an edge. To describe the degree of trust, we propose a metric named *user trust score*: for users  $u_i$  and  $u_j$ , the user trust score between them is denoted as  $\mathcal{T}(u_i, u_j)$ . Users with higher scores will be treated as trust users. The user trust score incorporates the information from two aspects: *user authority* and *sentiment similarity*. Next we will discuss the details about them.

User authority describes how a user is given attention to in the social network. A user with high authority can be regarded as the “opinion leader” in a specific field. If a user pays more attention to opinion leaders, it definitely means that he is interested in a particular topic and shares similar opinions with that user. So opinion leaders should be assigned higher degrees of trust. With the help of user authority, it is easy to find users with common interests.

To quantify user authority, for each user  $u_i \in U$ , we assign a user authority score denoted as  $\mathcal{A}(u_i)$ . According to above discussion, an opinion leader with more followers has higher authority, at the same time, followers will also contribute to the authority of opinion leader. Here in the user trust network, the in-degree of a node is the number of followers of a user; while the out-degree of node can be regarded as the influence a user has on other users. Then we can adopt the principle of *PageRank* to calculate the user authority score as shown in Eq. 1.

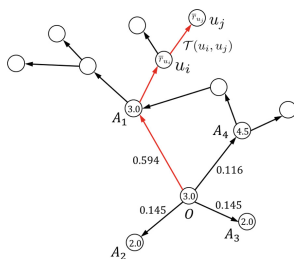
$$\mathcal{A}(u_i) = \frac{1 - \alpha}{|U|} + \alpha \left( \sum_{u \in N(u_i)} \frac{\mathcal{A}(u)}{L(u)} \right) \quad (1)$$

where  $N(u_i)$  denotes the set of nodes that have an edge pointed to  $u_i$ ,  $L(u_i)$  denotes the out-degree of node  $U_i$  and  $\alpha$  is the damping factor ( $\alpha \in (0, 1)$ ).

The sentiment similarity describes the similarity of sentiment orientation between users. For two users  $u_i$  and  $u_j$ , we use  $\mathcal{S}(u_i, u_j)$  to denote the sentiment similarity score between them. This score is evaluated by the average rating difference between users. And a lower difference means more similar rating orientation. With the help of sentiment similarity scores, we can find the candidate users who hold similar sentiment orientation. Equation 2 shows the sentiment similarity of users.

$$\mathcal{S}(u_i, u_j) = \frac{1}{\|\bar{r}_{u_i} - \bar{r}_{u_j}\| + 1} \quad (2)$$

where  $\bar{r}_{u_i}$  and  $\bar{r}_{u_j}$  are the average rating of user  $u_i$  and  $u_j$ , respectively. From this equation, we can see that the similarity between users is higher when their rating difference is lower.



**Fig. 2.** Toy example of user trust network

Finally we should take both user authority and sentiment similarity into consideration when deciding the user trust score. Intuitively if a user follows a high-authority user, they should be in the same user sequence. But if the sentiment similarity between them is high, putting them together might lead to some deviations in the result. Therefore, given two users  $u_i$  and  $u_j$ , we propose a hybrid scoring function by combining user authority and sentiment similarity scores of them. The way to calculate  $\mathcal{T}(u_i, u_j)$  is shown in Eq. 3.

$$\mathcal{T}(u_i, u_j) = \frac{\mathcal{A}(u_j) \cdot \mathcal{S}(u_i, u_j)}{\sum_{u \in N(u_i)} \mathcal{A}(u) \cdot \mathcal{S}(u_i, u)} \quad (3)$$

We take the user trust network shown in Fig. 2 as an example to illustrate the user trust scoring of node  $O$ . Suppose  $\bar{r}_O = \bar{r}_{A_1} = 3.0$ ,  $\bar{r}_{A_2} = \bar{r}_{A_3} = 2.0$ ,  $\bar{r}_{A_4} = 4.5$ . As shown in Eq. 2, the sentiment similarity between node  $O$  and its adjacent node  $A_i$  can be calculated as  $\mathcal{S}(O, A_1) = 1$ ,  $\mathcal{S}(O, A_2) = \mathcal{S}(O, A_3) = 0.5$ ,  $\mathcal{S}(O, A_4) = 0.4$ . With the input of the graph structure (adjacency list), we get the authority score  $\mathcal{A}(u)$  of every user, namely,  $\mathcal{A}(A_1) = 0.115$ ,  $\mathcal{A}(A_2) = \mathcal{A}(A_3) = \mathcal{A}(A_4) = 0.056$ . According to Eq. 3, we normalize  $\mathcal{A}(A_i) \cdot \mathcal{S}(O, A_i)$  to get the trust scores  $\mathcal{T}(O, A_i)$  of adjacent users as  $\mathcal{T}(O, A_1) = 0.594$ ,  $\mathcal{T}(O, A_2) = \mathcal{T}(O, A_3) = 0.145$ ,  $\mathcal{T}(O, A_4) = 0.116$ , which are used as the weight of the graph edge to generate user relation sequence from  $O$ .



### 3.3 User-Trust Random Walk Algorithm

Based on above user network, we can construct the user sequences by applying a random walk algorithm on the network structure. In order to include richer information in the user document, we want to include as many users in the sequence as possible. However, if a sequence of users is too long, the trust score will become pretty low, which means a rather low sentiment consistency. To make a trade-off between above factors, we set a stop probability  $\phi_{u_i, u_j, k}$  for the random walk algorithm as is shown in Eq. 4. It indicates the probability that  $u_i$  stops at  $u_j$  after  $k$  steps. We also record the maximum number of steps to make sure our algorithm could terminate. Each time when the stop condition is satisfied, we obtain a user relation sequence.

$$\phi_{u_i, u_j, k} = \frac{1}{1 + e^{-\frac{k}{2}}} \cdot \frac{\|\bar{r}_{u_i} - \bar{r}_{u_j}\|}{C} \quad (4)$$

where  $k$  is the step number,  $\bar{r}_{u_i}$  is the average rating of  $u_i$  and  $C$  is the number of classification categories.

**Input:** User set  $U$ , user trust network  $\mathcal{N}$ , maximum step length  $n$

**Output:** Random walk sequence for all users

```

1 for  $u \in U$  do
2    $k = 0, \phi = 0, u_m = u;$ 
3   Add  $u$  to its own user sequence;
4   while  $k < n$  and  $\text{rand}(0, 1) \geq \phi$  do
5     if  $u_m$  has adjacent users then
6       Calculate the trust score of  $u_m$  for  $N(u_m)$ ;
7       Sample the adjacent user in  $N(u_m)$  to  $u_{tmp}$ ;
8       Add  $u_{tmp}$  to the user sequence of  $u$ ;
9        $k = k + 1, u_m = u_{tmp}$ ;
10      Update stop probability  $\phi$ ;
11    end
12  end
13 end

```

**Algorithm 1.** Random walk on user trust network

Algorithm 1 shows the process of random walk. For all users in the network, we first initialize their sequences by involving themselves. Then if a user  $u_m$  has adjacent users, we will calculate the trust scores between  $u_m$  and all the adjacent users using Eq. 4. Next we perform a weighted sampling on the adjacent users, add the samples to  $u_m$ 's sequence and update the stop probability. We perform above computation iteratively until reaching the maximum step or meeting the stop probability.

## 4 Deep Learning Based Personalized Sentiment Classification

### 4.1 Overall Architecture

In this section, we proposed the deep neural network model SRPNN for personalized sentiment classification as is shown in Fig. 3. This model consists of two subnetworks: the left branch captures the features from semantics of review texts. The right branch is a CNN that models the user relations generated by the user trust network introduced in Sect. 3. Both branches consist of three layers: Word Representation Layer, Sentence Representation Layer and Document Representation Layer.

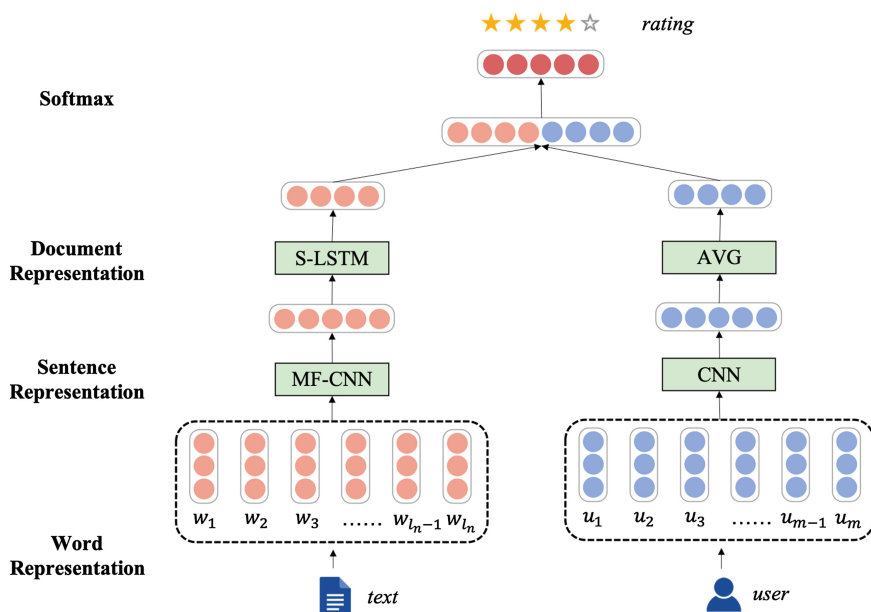


Fig. 3. The architecture of SRPNN model

### 4.2 Representation Learning for Text Document

We first introduce the left branch of our model which aims at learning the representation of review texts in Fig. 4. Here the first question we need to answer is how to generate the representative vector of a document. The semantics of a document can be obtained from the meanings of its sentences and the rules to compose words into a sentence. Following this routine, we can model the semantics of a document in two steps: we first generate the representation of

a sentence from the word embeddings. Afterwards, we composite the document representation with the embedding matrix of sentences.

For sentence level representation, we adopt the CNN model with multiple *filters* (MF-CNN) to extract the feature vectors from words. In the Word Representation layer, we transform words into representative vectors according to pretrained word embedding. Here  $[w_1, w_2, \dots, w_{l_n}]$  is the word sequence where  $l_n$  is the sentence length. Next we generate local features from the sequence of word embedding using convolution layers. To represent the sentence, we extract unigram, bigram and trigram features from the sentence. We can do it with the *filters* in the convolutional layer: we use multiple convolutional filters with different window sizes as  $l_c = 1, 2, 3$  to generate sentence representation. Then the input sequence of the convolution layer is  $I_c = [e_i, e_{i+1}, \dots, e_{i+l_c-1}]$  ( $i \in [1, l_n - l_c + 1], I_c \in \mathbb{R}^{d \times l_c}$ ). The convolution layer has the following linear transformation:

$$O_c = W_c \cdot I_c + b_c \tag{5}$$

where  $W_c \in \mathbb{R}^{l_{oc} \times d \times l_c}$ ,  $b_c \in \mathbb{R}^{l_{oc}}$  are parameters to be learned,  $l_{oc}$  denotes the output dimension of linear convolution. Upon the convolution layers, we adopt average pooling operation to aggregate  $l_n - l_c + 1$  local features. We also apply *tanh* function to develop the non-linear transformation in the hidden layers. Finally, we obtain three feature vectors with size  $l_{oc}$ . We use the average of above three vectors as the feature vector of sentence.

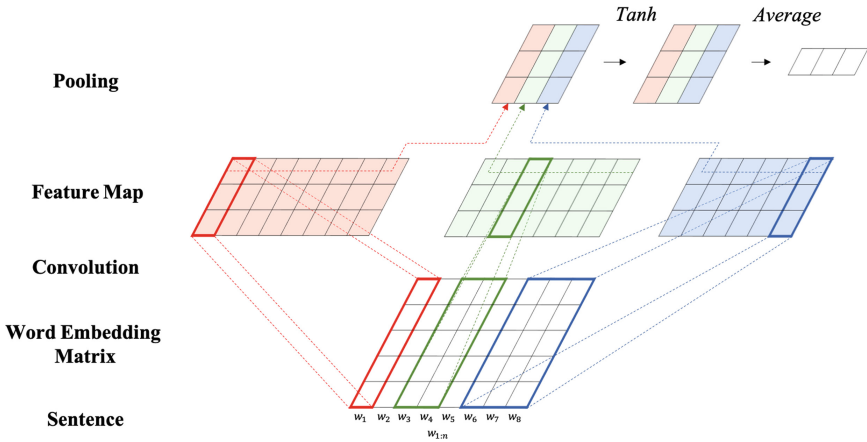


Fig. 4. Learning sentence representation with MF-CNN

To learn the document representation, we adopt the simplified LSTM (S-LSTM) model to generate the sentence-document vector. In this process, we hope to keep as much information of all sentences as possible. So compared with standard LSTM, we discard the output gate and replace the new state  $C_t$  with the origin candidate  $\tilde{C}_t$  to simplify LSTM, which is similar to the

Gated Recurrent Neural Network (GRNN) model. The gating mechanism of S-LSTM model is shown from Eqs. (6)–(9).

$$i_t = \text{sigmoid}(W_i \cdot [s_t; h_{t-1}] + b_i) \quad (6)$$

$$f_t = \text{sigmoid}(W_f \cdot [s_t; h_{t-1}] + b_f) \quad (7)$$

$$g_t = \text{tanh}(W_g \cdot [s_t; h_{t-1}] + b_g) \quad (8)$$

$$h_t = \text{tanh}(i_t \odot g_t + f_t \odot h_{t-1}) \quad (9)$$

where  $s_t$  is the sentence vector at time  $t$ ,  $W_i$ ,  $W_f$ ,  $W_g$  are the weight matrices.  $b_i$ ,  $b_f$ ,  $b_g$  are the offset vectors,  $\odot$  denotes the element-wise multiplication,  $i_t$  is the input gate,  $f_t$  is the output gate,  $g_t$  is the new state and  $h_t$  is the output at time  $t$ . We regard the output of simplified LSTM as the feature vector of the document.

Figure 5 shows the simplified LSTM to learn the document representation from sentences. The input of each time  $t_i$  is the sentence vector  $s_i$  and the latent output  $h_{i-1}$  in the previous time unit; and the output is  $h_i$  correspondingly. We get the output of each time unit iteratively and finally obtain  $h_n$  as the output of this model.

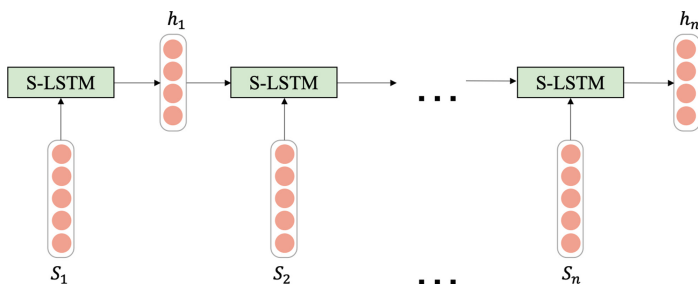


Fig. 5. The simplified LSTM for learning document representation

### 4.3 Representation Learning for User Document

We then present the right branch of our model to learn the user documents. The representation of user documents is also obtained in two steps, the same as text document: first learning the user sequence and then the document. We generate the user document in the following way: each user in the user network is assigned an identifier. An identifier is regarded as a “word”. As we have generated the sequences of users using random walk algorithm, we regard each user sequence as a “sentence”. All the user sequences of the same user construct that user’s document.

Similar to the text document, we also learn the representation of user document using the CNN model. However, as the sentences in user document are generated through executing the random walk algorithm multiple times, the weights of all the sentences are equivalent. Unlike the text reviews, words in

the user document are not closely related to each other in the contexts. Therefore we only use one window size of filter in convolution layers. We set it as 5 empirically. For learning the document representation, since the sentence order of user documents does not provide semantic information, we do not use LSTM to capture high-level features. Instead we just average the sentence vectors to generate the document representation.

#### 4.4 Output of the Model

As the feature vectors are obtained from both text reviews and user documents, we then generate the joint features by concatenating them, which has been shown in Fig. 3. Then we feed it into a fully connected layer to predict the sentiment label. We adopt softmax function to learn the probability of each classification label. In the training process, we use cross-entropy loss as our loss function. And correspondingly the objective function is denoted in Eq. 10.

$$loss = \sum_{d \in T} \sum_{i=1}^C P_i^g(d) \cdot \log(P_i(d)) \quad (10)$$

where  $T$  is the training set,  $d$  is a document in the training set,  $C$  is the number of classification categories,  $P_i^g(d)$  denotes whether document  $d$  belongs to class  $i$  (1 when true or 0 when false),  $P_i(d)$  is the probability of prediction for document  $d$  with class  $i$ . We use the Adagrad algorithm to optimize the training process and back-propagation to learn the model parameters. The learning rate is set as 0.03.

**Table 1.** The statistics of datasets

Item	Twitter_1000	Twitter	Yelp
# of posts	76517	1446557	1569264
# of users	1000	596714	366715
# of words	79266	805762	742875
# of posts per user	76.52	2.42	4.28
# of sentence per post	2.86	2.78	9.99
# of words per post	17	16.53	145.02
# of following people per user	8.24	32.97	7.55
sentiment distribution	0.37/0.63	0.5/0.5	0.1/0.09/0.14/0.30/0.37

## 5 Evaluation

### 5.1 Data Observations

**Datasets.** We conduct an extensive set of experiments on two widely used datasets: Twitter and Yelp. Table 1 describes the statistical information of the datasets. Twitter is obtained from the Sentiment140 dataset<sup>1</sup> and Twitter user network [8]. Yelp is obtained from Yelp Dataset Challenge in 2015<sup>2</sup>. The training, validation and test set are constructed by randomly splitting data from a user in the portion of (80/10/10). These two datasets are rather sparse. Following the previous study [33], we also build the dense dataset Twitter\_1000 from the top 1000 users with most tweets. The number of classification labels is two for Twitter and five for Yelp respectively.

Figure 6 shows the distribution of post number. We can see that the post number obeys the long-tailed distribution. This can further prove the sparsity of our datasets. We also describe the word frequency and follower number of each datasets in Fig. 7. The distribution of word frequency and follower number can demonstrate the property of text and user respectively. We can see that Twitter and Yelp datasets perform similarly in both word frequency and follower number. Even in Twitter\_1000 we have filtered users with highest number of tweets, we can still get the similar distributions. This demonstrates that we can use a unified model to learn the representation of the user and document.

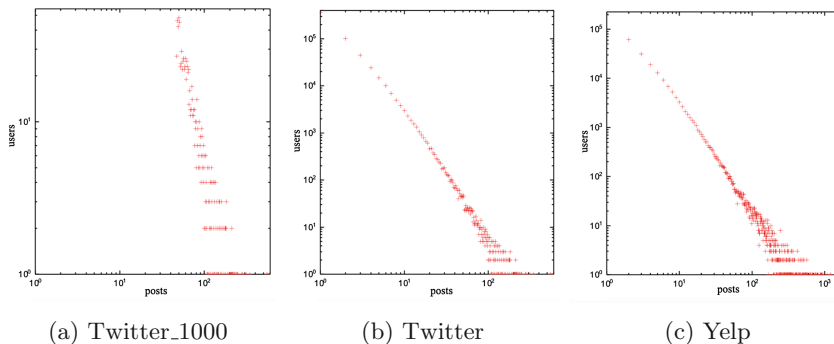


Fig. 6. Post distribution of datasets

**Experiment Setup.** Similar to previous studies [25, 33], we use Accuracy and Root Mean Square Error (RMSE) to measure the overall sentiment classification performance.

<sup>1</sup> <http://help.sentiment140.com>.

<sup>2</sup> [https://www.yelp.com/dataset\\_challenge](https://www.yelp.com/dataset_challenge).

$$\text{Accuracy} = \frac{1}{N} \sum_{i=1}^N 1(\hat{r}_i == r_i) \quad (11)$$

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^N (\hat{r}_i - r_i)^2}{N}} \quad (12)$$

The RMSE denotes the divergence between predicted sentiment rating ( $\hat{r}_i$ ) and ground truth rating ( $r_i$ ).

Next we introduce the setting of hyper parameters. For learning the user trust network, we set the random walk step as 10, and every user has 10 random walk sequences. We use the pre-trained word2vec embedding with 200 dimensions. The number of filters of all convolution layers are set as 100. The output dimension of LSTM is set as 60. The dimension of hidden layers in both branches is 50.

## 5.2 Baseline Methods

We compare SRPNN with following state-of-the-art baselines of sentiment classification domain, including two feature-based and three deep learning methods:

**SVM+N-gram.** Following many previous studies [23–25, 27, 33], we use uni-gram, bigram and trigram as features to train a SVM classifier as the baseline method.

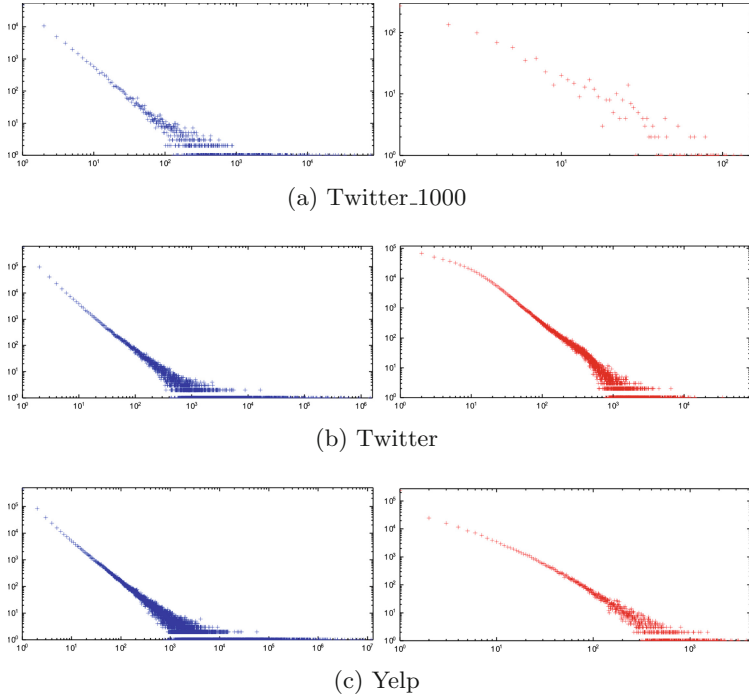
**PMSC.** Personalized Microblog Sentiment Classification (PMSC) [33] is a feature-based method assigning personalized weight parameter for each user. As it only supports binary classification, we did not report its results on Yelp dataset.

**ParaVec.** ParaVec [9] regards each document as a paragraph and learns the paragraph representation. We use the Paravec features and adopt SVM as the classifier to develop a supervised learning.

**GRNN.** Gated Recurrent Neural Network (GRNN) takes simplified LSTM to classify sentiment polarity by [24]. They study the document modeling methods with deep learning and get significantly better performance than existing approaches.

**UPNN.** User and Product Neural Network (UPNN) is a personalized sentiment classification model based on convolutional neural network [25].

We obtained the source code of above methods from the authors. And for all the methods, we tune the hyper parameters according to original papers and report the best results we achieved.



**Fig. 7.** Word frequency (left) and follower number (right) of datasets

**Table 2.** Compare with state-of-the-art methods

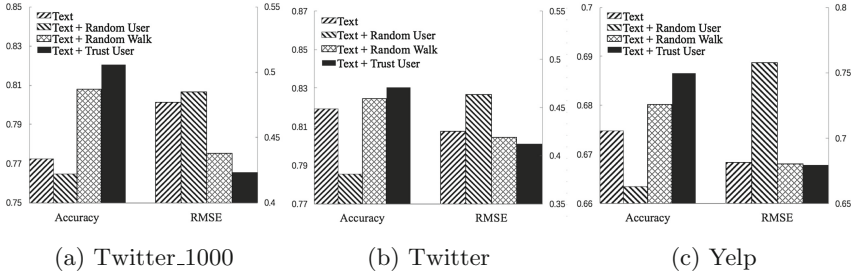
Model		Twitter_1000		Twitter		Yelp	
		Accuracy	RMSE	Accuracy	RMSE	Accuracy	RMSE
Feature-based methods	SVM+unigram	0.7693	0.4804	0.7883	0.4601	0.6235	0.8356
	SVM+bigram	0.7888	0.4596	0.8067	0.4396	0.6357	0.7762
	SVM+trigram	0.7940	0.4538	0.8169	0.4279	0.6436	0.7580
	PMSC	0.8211	0.4230	–	–	–	–
Deep learning methods	ParaVec	0.7495	0.5005	0.7650	0.4847	0.6016	0.8965
	GRNN	0.7724	0.4770	0.8193	0.4250	0.6748	0.6812
	UPNN	<b>0.8218</b>	<b>0.4222</b>	0.8192	0.4252	0.6475	0.7576
	SRPNN	0.8205	0.4237	<b>0.8304</b>	<b>0.4118</b>	<b>0.6865</b>	<b>0.6793</b>

### 5.3 Model Comparisons and Analysis

We first compare our SRPNN with state-of-the-art methods. Table 2 shows our experimental results. Note the PMSC method runs out of memory for Twitter datasets, so we cannot report its performance here. We can see that SRPNN outperforms all the other baselines on Twitter and Yelp datasets. The reason is



that when faced with sparse data, other methods fail to capture enough informative features. Although UPNN can take advantage of products information, the benefit is limited due to the problem of data sparsity. The observation that our SRPNN model outperforms GRNN can further demonstrate the effect of user trust network towards personalized sentiment classification: GRNN only focuses on text features with a simplified LSTM, while SRPNN integrates features from both text reviews and user trust network.



**Fig. 8.** Effect of proposed techniques

We then evaluate the effectiveness of our user-trust random walk algorithm by changing the settings of user document. We compared it with 3 baseline methods: *Text* does not include user document; *Text+Random User* uses randomly generated user document; *Text+Random Walk* adopts random walk algorithm on the unweighted directed graph generated from user following relations; *Text+Trust User* is our proposed method. The results are shown in Fig. 8. We can see that *Text+Trust User* achieves the best results. At the same time, *Text+Random Walk* ranks second as it involves user relation information. It is worth noting that *Text+Random User* performs worst. The reason could be that randomly generated user sequence contributes nothing but noise in the user document. It indicates that our user trust network can provide important information of sentiment.

Finally we analyze the result on Twitter\_1000. We can see that SRPNN gets very similar results with state-of-the-art methods. This is reasonable because personalized modeling of a user has already been well supported by original data. So there is no data sparsity. As both PMSC and UPNN directly model users, they will have better performance on dense dataset. However, in the real applications such as social media and review rating of products, datasets are often very sparse. Therefore, although SRPNN does not outperform PMSC and UPNN on Twitter\_1000, it is still necessary in real-world scenarios.

## 6 Conclusion

In this paper, we propose Social Relation Powered Neural Network (SRPNN), a deep learning based model for document-level sentiment classification. We propose a random walk algorithm to obtain the sequences of users with user-sentiment consistency so as to generate the user document. We then jointly learn the representation of text and user document. Experimental results on two public datasets demonstrate the effectiveness of our proposed methods.

**Acknowledgment.** This work was supported by NSFC (91646202), National Key R&D Program of China (SQ2018YFB140235), and the 1000-Talent program.

## References

1. Chen, H., Sun, M., Tu, C., Lin, Y., Liu, Z.: Neural sentiment classification with user and product attention. In: EMNLP, pp. 1650–1659 (2016)
2. Dong, L., Wei, F., Zhou, M., Xu, K.: Adaptive multi-compositionality for recursive neural models with applications to sentiment analysis. In: AAAI, pp. 1537–1543 (2014)
3. Fouss, F., Pirotte, A., Renders, J., Saerens, M.: Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Trans. Knowl. Data Eng.* **19**(3), 355–369 (2007)
4. Golbeck, J.: Computing and applying trust in web-based social networks. Ph.D. thesis, University of Maryland, College Park (2005)
5. Hu, X., Tang, L., Tang, J., Liu, H.: Exploiting social relations for sentiment analysis in microblogging. In: WSDM, pp. 537–546 (2013)
6. Jamali, M., Ester, M.: TrustWalker: a random walk model for combining trust-based and item-based recommendation. In: KDD, pp. 397–406 (2009)
7. Kiritchenko, S., Zhu, X., Mohammad, S.M.: Sentiment analysis of short informal texts. *J. Artif. Intell. Res.* **50**, 723–762 (2014)
8. Kwak, H., Lee, C., Park, H., Moon, S.B.: What is Twitter, a social network or a news media? In: WWW, pp. 591–600 (2010)
9. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. In: ICML, pp. 1188–1196 (2014)
10. Li, F., Liu, N.N., Jin, H., Zhao, K., Yang, Q., Zhu, X.: Incorporating reviewer and product information for review rating prediction. In: IJCAI, pp. 1820–1825 (2011)
11. Li, Z., Wei, Y., Zhang, Y., Yang, Q.: Hierarchical attention transfer network for cross-domain sentiment classification. In: AAAI, pp. 5852–5859. AAAI Press (2018)
12. Liu, K., Li, W., Guo, M.: Emoticon smoothed language models for Twitter sentiment analysis. In: AAAI (2012)
13. Luo, L., et al.: Beyond polarity: interpretable financial sentiment analysis with hierarchical query-driven attention. In: IJCAI, pp. 4244–4250 (2018)
14. Massa, P., Avesani, P.: Trust-aware recommender systems. In: RecSys, pp. 17–24 (2007)
15. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS, pp. 3111–3119 (2013)

16. Mishra, A., Dey, K., Bhattacharyya, P.: Learning cognitive features from gaze data for sentiment and sarcasm classification using convolutional neural network. In: *ACL*, pp. 377–387 (2017)
17. Mnih, V., Heess, N., Graves, A., Kavukcuoglu, K.: Recurrent models of visual attention. In: *NIPS*, pp. 2204–2212 (2014)
18. Pang, B., Lee, L.: Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In: *ACL* (2005)
19. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. In: *KDD*, pp. 701–710 (2014)
20. Qian, Q., Huang, M., Lei, J., Zhu, X.: Linguistically regularized LSTM for sentiment classification. In: *ACL*, pp. 1679–1689 (2017)
21. Qu, L., Ifrim, G., Weikum, G.: The bag-of-opinions method for review rating prediction from sparse text patterns. In: *COLING*, pp. 913–921 (2010)
22. Socher, R., et al.: Recursive deep models for semantic compositionality over a sentiment treebank. In: *EMNLP*, pp. 1631–1642 (2013)
23. Song, K., Feng, S., Gao, W., Wang, D., Yu, G., Wong, K.: Personalized sentiment classification based on latent individuality of microblog users. In: *IJCAI*, pp. 2277–2283 (2015)
24. Tang, D., Qin, B., Liu, T.: Document modeling with gated recurrent neural network for sentiment classification. In: *EMNLP*, pp. 1422–1432. The Association for Computational Linguistics (2015)
25. Tang, D., Qin, B., Liu, T.: Learning semantic representations of users and products for document level sentiment classification. In: *ACL*, pp. 1014–1023 (2015)
26. Tang, D., Qin, B., Liu, T.: Aspect level sentiment classification with deep memory network. In: *EMNLP*, pp. 214–224 (2016)
27. Tang, D., Qin, B., Liu, T., Yang, Y.: User modeling with neural network for review rating prediction. In: *IJCAI*, pp. 1340–1346 (2015)
28. Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., Qin, B.: Learning sentiment-specific word embedding for Twitter sentiment classification. In: *ACL*, pp. 1555–1565 (2014)
29. Wang, J., Lin, C., Li, M., Zaniolo, C.: An efficient sliding window approach for approximate entity extraction with synonyms. In: *EDBT* (2019)
30. Wang, J., Wang, Z., Zhang, D., Yan, J.: Combining knowledge with deep convolutional neural networks for short text classification. In: *IJCAI*, pp. 2915–2921 (2017)
31. Wang, W., Feng, S., Gao, W., Wang, D., Zhang, Y.: Personalized microblog sentiment classification via adversarial cross-lingual multi-task learning. In: *EMNLP*, pp. 338–348. Association for Computational Linguistics (2018)
32. Wasserman, S., Faust, K.: *Social Network Analysis: Methods and Applications*, vol. 8. Cambridge University Press, Cambridge (1994)
33. Wu, F., Huang, Y.: Personalized microblog sentiment classification via multi-task learning. In: *AAAI*, pp. 3059–3065 (2016)
34. Wu, Z., Dai, X., Yin, C., Huang, S., Chen, J.: Improving review representations with user attention and product attention for sentiment classification. In: *AAAI*, pp. 5989–5996. AAAI Press (2018)
35. Zhang, Y., Wu, J., Wang, J., Xing, C.: A transformation-based framework for KNN set similarity search. *IEEE Trans. Knowl. Data Eng.* (2019)
36. Zhao, K., et al.: Modeling patient visit using electronic medical records for cost profile estimation. In: Pei, J., Manolopoulos, Y., Sadiq, S., Li, J. (eds.) *DASFAA 2018*. LNCS, vol. 10828, pp. 20–36. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-91458-9\\_2](https://doi.org/10.1007/978-3-319-91458-9_2)

37. Zhao, Z., Lu, H., Cai, D., He, X., Zhuang, Y.: Microblog sentiment classification via recurrent random walk network learning. In: IJCAI, pp. 3532–3538. ijcai.org (2017)
38. Zhu, L., Galstyan, A., Cheng, J., Lerman, K.: Tripartite graph clustering for dynamic sentiment analysis on social media. In: SIGMOD, pp. 1531–1542 (2014)



# Reinforcement Learning to Diversify Top-N Recommendation

Lixin Zou<sup>1</sup>(✉), Long Xia<sup>2</sup>, Zhuoye Ding<sup>2</sup>, Dawei Yin<sup>2</sup>, Jiaxing Song<sup>1</sup>,  
and Weidong Liu<sup>1</sup>

<sup>1</sup> Department of Computer Science and Technology,  
Tsinghua University, Beijing, China

{zoulx15, jxsong, liuwd}@mails.tsinghua.edu.cn

<sup>2</sup> Data Science Lab, JD.com, Beijing, China

{xialong, dingzhuoye}@jd.com, yindawei@acm.org

**Abstract.** In this paper, we study how to recommend both accurate and diverse top-N recommendation, which is a typical instance of the maximum coverage problem. Traditional approaches are to treat the process of constructing the recommendation list as a problem of greedy sequential items selection, which are inevitably sub-optimal. In this paper, we propose a reinforcement learning and neural networks based framework – **Diversify** top-N Recommendation with **F**ast **M**onte **C**arlo **T**ree **S**earch (Div-FMCTS) – to optimize the diverse top-N recommendations in a global view. The learning of Div-FMCTS consists of two stages: (1) searching for better recommendation with MCTS; (2) generalizing those plans with the policy and value neural networks. Due to the difficulty of searching over extremely large item permutations, we propose two approaches to speeding up the training process. The first approach is pruning the branches of the search tree by the structure information of the optimal recommendations. The second approach is searching over a randomly chosen small subset of items to quickly harvest the fruits of searching in the generalization with neural networks. Its effectiveness has been proved both empirically and theoretically. Extensive experiments on four benchmark datasets have demonstrated the superiority of Div-FMCTS over state-of-the-art methods.

**Keywords:** Recommender system · Recommendation diversity · Monte Carlo Tree Search

## 1 Introduction

With the rapid growth of the Internet, the recommender system has become an indispensable part for many companies (e.g. Amazon<sup>1</sup>, Netflix<sup>2</sup>, JD.com<sup>3</sup>) to

---

L. Zou—Work performed during an internship at JD.com.

<sup>1</sup> <https://www.amazon.com>.

<sup>2</sup> <https://www.netflix.com>.

<sup>3</sup> <https://www.jd.com>.

© Springer Nature Switzerland AG 2019

G. Li et al. (Eds.): DASFAA 2019, LNCS 11447, pp. 104–120, 2019.

[https://doi.org/10.1007/978-3-030-18579-4\\_7](https://doi.org/10.1007/978-3-030-18579-4_7)

overcome the information overload and to help customers find products. As a basic service, recommender system aims at filtering out the unattractive items and generating item recommendations in favor of user preferences. To cater to users, the recommender system is designed to find best-fitted products for users (e.g. highest rated movies, likest cellphone), which results in researches focusing on improving the accuracy of recommendations. However, the most accurate recommendations are sometimes not the recommendations that are most useful to users [22, 33]. Fully exploiting learned user preferences without exploration of probably liking items might result in performance deterioration and falling into the cycle of recommending same items, such as always recommending cellphones if a new user only clicked a cellphone, which would disappoint users and cause the loss of users. Improving the recommendation diversity has been recognized as an effective way to alleviate this issue because it can broaden users' horizon and help users find new interesting items. Additionally, the platform can improve both their performance and users' satisfaction, which is a win-win situation.

During the last decade, various diversity-enhancing methods have been developed to increase diversity while maintaining the accuracy [1–3, 5, 41]. The previous methods for diversity improvement can be roughly divided into two categories: point-wise and list-wise approaches. The point-wise approaches usually involve two phases: generating the candidate items with the highest accuracy and re-ranking the items by heuristic methods [1], such as re-ranking with popularity [2]. For list-wise approach, it directly trains a supervised learning model to generate the recommendation lists with high accuracy and diversity [5]. However, the ground truth training labels are obtained by greedy selection. All in all, the existing approaches are treating the process of constructing the recommendation list as a problem of greedy sequential items selection. At each step, the algorithm iteratively selects the item with the highest marginal gain (which usually takes both accuracy and diversity into account) with respect to the selected items. However, the diversity of an item depends on the other recommended items, selecting an optimal ranking of items is a typical instance of the maximum coverage problem, a classical NP-hard problem in computational complexity theory [11]. Therefore, the recommendation lists produced by greedy items selection are inevitably sub-optimal. In general, the recommendation algorithm needs to explore the whole candidate item space, if the optimal recommendation list is mandatory. However, it is usually infeasible in real recommender systems as the huge space of possible recommendations: for selecting  $N(N \sim 10^1)$  items from  $K(K > 10^4)$  candidate items there exist  $\frac{K!}{N!} (\gg 10^{10})$  different recommendation permutations. In this way, there is an urgent need for an algorithm suitable for recommendation diversity task that can efficiently search for optimal results in large spaces.

In this work, we will investigate to obtain accurate and diverse top-N recommendation in a sequential decision-making manner. To avoid local optimal solution and learn the optimal global policy efficiently, we propose a novel reinforcement learning and neural network based approach named **Diversify recommendation with Fast Monte Carlo Tree Search (Div-FMCTS)**.

Specifically, Div-FMCTS formulates the generation of recommendation as a finite-horizon MDP and finds the optimal policy by iterating between two procedure: (1) searching the space of item permutations to find optimal top-N recommendations; (2) generalizing those searching results with neural networks. In searching stage, Monte Carlo Tree Search (MCTS) is proposed to heuristically search for the optimal recommendations. However, searching with MCTS is slow due to the extremely large size of candidate items. In this work, we propose two approaches to deal with large items set. The first approach, called structure pruning, uses the structure information of optimal recommendations to narrow down the search space of MCTS. The second approach, called problem decomposition, searches over a small randomly chosen subset of candidate items to quickly harvest the fruits of MCTS and generate more training sets for the imitation learning in the second stage. To prove its validity, we theoretically and empirically verify that diversifying recommendations can be equivalently solved by searching over many subset candidate item set. In the generalizing stage, a two-head GRU with a factorized similarity model [37] is proposed to approximate the searching results and estimate the goodness of current recommendations by tracking both user’s temporary and intrinsic interests. Finally, extensive experiments on four benchmark MovieLens datasets show that Div-FMCTS can recommend diverse items at the same time maintaining high accuracy, which can not be achieved by either traditional methods or the state-of-the-art methods [2, 3, 9, 17, 25, 26].

The rest of the paper is organized as follows. Section 2 discusses the formulation of diversify top-N recommendation and its difficulty. Section 3 describes our proposed model and learning algorithm in detail. In Sect. 4, we propose how to speed up the tree search process. We discuss related work on the diverse top-N recommendation in Sect. 5. Experimental results for the analysis and performance comparisons are presented in Sect. 6. We conclude with a summary of this work in Sect. 7.

## 2 Diverse Top-N Recommendation as MDP

### 2.1 Diverse Top-N Recommendation

Assuming that there are a set of users  $\mathcal{U}$  and a set of items  $\mathcal{I}$ . For each item  $i \in \mathcal{I}$ , it lies on a set of topics/categories  $\mathcal{Z}_i = \{\zeta_{i_j}\}_{j=1}^Z$ , where  $\zeta_{i_j}$  is one topic of item  $i$  and  $Z$  is the number of associated topics, usually ranging from 1 to 5. Given a user’s rated history  $\mathcal{O}_u = \{(i_{o_j}, r_{o_j})\}_{j=1}^O$  where  $r_{o_j}$  are numeric ratings, we are concerned with finding ranked N-items  $\mathbf{i}_\nu = [i_{\nu_t}]_{t=1}^N$  from not being rated set  $\mathcal{C}_u = \mathcal{I} - \{i_{o_j}\}_{j=1}^O$  that maximize the trade-off between accurate and diverse metrics as

$$\max_{\{\mathbf{i}_\nu\}_{t=1}^N \subset \mathcal{C}_u} 2 \frac{f(\mathbf{i}_\nu) \times g(\mathbf{i}_\nu)}{f(\mathbf{i}_\nu) + g(\mathbf{i}_\nu)}, \quad (1)$$

where  $f, g : Y \rightarrow R$  are the metrics on recommendation accuracy and diversity respectively, e.g. NDCG,  $\alpha$ -NDCG, Pairwise Accuracy Metric (PA), Normalized

Topic Coverage (NTC) [5, 6, 13].  $2 \frac{f(\cdot)g(\cdot)}{f(\cdot)+g(\cdot)}$  is the F-measure between accuracy and diversity, which is being widely used as the evaluation metric on recommendation task [5].

Theoretically, recommendation diversity can be naturally stated as a bi-criterion optimization problem, and it is NP-hard [3]. To comprehend its difficulty, Fig. 1 shows a toy example, which includes the candidate set, the optimal ranking and greedy ranking of top-3 recommendations. The greedy solution, selecting the item that maximizing the relevance and diversity gain, is different with optimal recommendations. In practice, most previous approaches on recommendation diversity are based on greedy approximation, which sequentially selects a ‘local-best’ item from the remanent candidate set [8], which inevitably is the suboptimal recommendations. Therefore, it is urgent to view recommendation as a sequential selection process and to optimize the ranking in a global view.

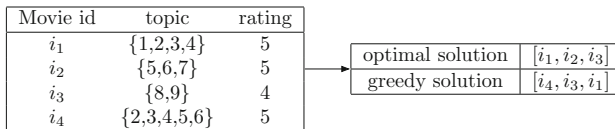


Fig. 1. A toy example of top-3 recommendation.

## 2.2 MDP Formulation of Diverse Recommendation

In this work, we formulate the diverse top-N recommendation as a Markov Decision Process (MDP) and optimize whole recommendations with reinforcement learning. A MDP is defined by  $M = \langle S, A, P, R, \gamma \rangle$ , where  $S$  is the state space,  $A$  is the action space,  $P : S \times A \times S \rightarrow \mathbb{R}$  is the transition function with  $p(s_{t+1}|s_t, a_t)$  being the probability of seeing state  $s_{t+1}$  after taking action  $a_t$  at  $s_t$ ,  $R : S \times A \rightarrow \mathbb{R}$  is the mean reward function with  $r(s, a)$  being the immediate goodness of  $(s, a)$ , and  $\gamma \in [0, 1]$  is the discount factor. We design the state at time step  $t$  as a tuple  $s_t = \{\mathbf{i}_{\nu_{<t}}, u, \mathcal{O}_u\}$  consisted of the user’s rated history  $\mathcal{O}_u$ , user id  $u$  and previous  $t - 1$  recommendations  $\mathbf{i}_{\nu_{<t}} = [i_{\nu_1}, i_{\nu_2}, \dots, i_{\nu_{t-1}}]$ . Given  $s_t$ , the recommendation agent chooses a action, e.g. the recommendation  $i_{\nu_t}$ , and updates the state by appending  $i_{\nu_t}$  to  $\mathbf{i}_{\nu_{<t}}$ . Then, a reward  $r_t(s_t, i_{\nu_t})$  should be given for  $i_{\nu_t}$ . To be consistent with the metrics defined in Eq. (1), we consider the reward function as the accuracy and diversity gain deriving from recommending item  $i_{\nu_t}$  at position  $t$  as

$$r_t(s_t, i_{\nu_t}) = 2 \frac{G(i_{\nu_t}) \times \alpha - G(i_{\nu_t})}{\log_2(t+1)(G(i_{\nu_t}) + \alpha - G(i_{\nu_t}))}, \quad (2)$$

where  $r_t(s_t, i_{\nu_t})$  is the discounted F-measure of accuracy gain  $G(i_{\nu_t})$  and diversity gain  $\alpha - G(i_{\nu_t})$  on  $t$ -th recommendation.  $\frac{1}{\log_2(t+1)}$  is the discounted factor for the  $t$ -th recommendation, which has been widely used in metrics for recommendation evaluation [6, 13]. Because of the discount factor and the fact that top-N



recommendation is a finite-horizon MDP problem, there is no need for additional discount on the reward function and we set the discounted factor as  $\gamma = 1$ .

To maximize the reward, we usually learn a policy  $\pi : S \times A \rightarrow \mathbb{R}$ , which maps the state  $s_t$  to the probability of recommending  $i_{\nu_t}$  at time step  $t$ . With the policy  $\pi$ , the probability of recommending  $i_{\nu}$  can be modeled by the chain rule as,

$$p(\mathbf{i}_{\nu}) = \prod_{t=1}^N \pi(i_{\nu_t} | s_t). \quad (3)$$

We aim to learn a parameterized policy  $\pi_{\theta}$  that assigns highest probability  $p(\mathbf{i}_{\nu})$  to the recommendation  $\mathbf{i}_{\nu}^*$  with the biggest reward, e.g. maximizing the accuracy and diversity of top-N recommendations.

### 3 The Training Framework

To find the optimal recommendation policy  $\pi_*$ , we split the learning algorithm into two-stage iterative optimization problem: (1) given the parameterized policy  $\pi_{\theta}$ , searching for better policies  $\pi_e$  by using  $\pi_{\theta}$  as prior knowledge. In this work, we employ MCTS for finding the better policy  $\pi_e$ , which usually selects much stronger moves than the raw probabilities  $\pi_{\theta}$  [30]; (2) updating the  $\pi_{\theta}$  by minimizing the difference between  $\pi_e$  and  $\pi_{\theta}$ . Then, the updated policy  $\pi_{\theta}$  is used in the next iteration to make the  $\pi_e$  even stronger.

#### 3.1 The Architecture of Policy and Value Neural Networks

For the success of training a good policy, the architecture of policy function is important. Apart from the policy function, the value function is also parameterized in our neural network, which evaluates the goodness of following current policy. We employ a two-head architecture for reinforcement learning, which has two advantages: (1) training with the information from the expert policy and estimating value can effectively avoid the overfitting; (2) the parameterized value function is helpful for fast estimation of leaf nodes' value in the tree search phases. In the following, we first discuss how to model user's preferences in states. Then, based on user's preferences, the policy and value functions have been built.

*Preference Embedding.* The state  $s_t = \{\mathbf{i}_{\nu_{<t}}, u, \mathcal{O}_u\}$  contains user's historical information. We first computes state embedding as,

$$\mathbf{s}_t = \left[ \mathbf{u}^{\top}, GRU(\mathcal{O}_u, \mathbf{i}_{\nu_{<t}})^{\top} \right]^{\top}$$

where  $\mathbf{u}^{\top} \in \mathcal{R}^{d_u}$  is a latent factor used to embed user's intrinsic preference.  $GRU(\mathcal{O}_u, \mathbf{i}_{\nu_{<t}})$  is a GRU model to summary user's temporary preference on

first  $t - 1$  recommendation. The initial hidden state  $\mathbf{h}_0 = GRU(\mathcal{O}_u)$  is set by a factorized similarity model [37] as,

$$\mathbf{h}_0 = \sum_{(i_{o_j}, r_{o_j}) \in \mathcal{O}_u} r_{o_j} \mathbf{i}_{o_j} + \mathbf{b}_u$$

where  $\mathbf{b}_u \in \mathcal{R}^{d_H}$  is the bias term,  $\mathbf{i}_{o_j} \in \mathcal{R}^{d_H}$  is the embedding vector for item  $i_{o_j}$ .

*The Policy and Value Function.* Given the state embedding  $\mathbf{s}_t$ , the recommendation policy  $\pi_\theta$  and value function  $v_\theta$  are defined as

$$\begin{aligned} \pi_\theta(i_{\nu_t} | \mathbf{i}_{\nu_{<t}}, u, \mathcal{O}_u) &= \frac{\exp(\Gamma(i_{\nu_t} | \mathbf{s}_t))}{\sum_{i_j \in \mathcal{I}} \exp(\Gamma(i_j | \mathbf{s}_t))} \\ \Gamma(i_1, \dots, i_I | \mathbf{s}_t) &= W_s \mathbf{s}_t + \mathbf{b}_s \\ v_\theta(\mathbf{s}_t) &= \mathbf{V}^\top \mathbf{s}_t + b_v \end{aligned}$$

where  $W_s \in \mathcal{R}^{(d_U+d_H) \times d_I}$ ,  $\mathbf{V} \in \mathcal{R}^{d_U+d_H}$ ,  $\mathbf{b}_s \in \mathcal{R}^{d_I}$  and  $b_v$  are the weight and bias terms.  $\Gamma(i_j | \mathbf{s}_t)$  is the score for recommending item  $i_j$ .  $\pi_\theta$  is set as the softmax of score  $\Gamma(i_j | \mathbf{s}_t)$ .

### 3.2 Searching for the Expert Policy

In the first phases, we employ MCTS to search for an improved policy compared with current  $\pi_\theta$ . In the executing process, the MCTS planning process incrementally builds an asymmetric search tree that is guided in the most promising direction by an exploratory action-selection policy. This process usually consists of four phases: **selection**, **evaluation**, **expansion** and **backup** as

- **Selection:** At each node  $s_t$ , MCTS uses a selection policy to balance of exploitation of actions with high value estimates and exploration of actions with uncertain value estimates. To balance the trade-off of exploitation and exploration, a variant of PUCT is employed [30]. Specifically, an action  $i_{\nu_t}$  is selected at node  $s_t$  so as to maximize action value plus uncertain estimate:

$$i_{\nu_t} = \arg \max_i (Q(s_t, i) + c_{puct} P(s_t, i) \frac{\sqrt{\sum_{b \in \mathcal{C}_u} N(s_t, b)}}{1 + N(s_t, i)})$$

where  $Q(s_t, a)$  is the mean action-value,  $P(s_t, i) \frac{\sqrt{\sum_{b \in \mathcal{C}_u} N(s_t, b)}}{1 + N(s_t, i)}$  is the uncertain value estimate,  $P(s_t, i)$  is the prior probability of selecting that edge,  $N(s_t, i)$  is the visit count and  $c_{puct}$  is the constant determining the level of exploration.

- **Evaluation:** When the iteration reaches a leaf node  $s_t$ , the node  $v(s_t)$  is evaluated either with the value function  $v_\theta(s_t)$  or with the predefined performance measure if the node is the end of an episode and the human labels are available.

- **Expansion:** The leaf node  $s_t$  may be expanded. Each edge from the leaf position  $s_t$  is initialized as:  $P(s_t, i_{\nu_t}) = p(i_{\nu_t}|s_t)$ ,  $Q(s_t, i_{\nu_t}) = 0$ , and  $N(s_t, i_{\nu_t}) = 0$ . In this paper all of the available actions of  $s_t$  are expanded.
- **Backup:** Finally, we recursively back-up the results in the tree nodes. Denoting the current forward trace in the tree as  $\{s_1, i_1, s_2, \dots, s_t\}$ . For  $\forall (s_j, i_j) j < t$ , we recursively update the  $Q(s_j, i_j)$  as

$$Q(s_j, i_j) \leftarrow \frac{N(s_j, i_j) \times Q(s_j, i_j) + v(s_j)}{N(s_j) + 1}$$

$$v(s_j) \leftarrow v(s_{j+1}) + r_j(s_j, i_j).$$

This cycle of selection, evaluation, expansion and backup is repeated until the maximum iteration number has been reached. At this point, the best action is been chosen by selecting the action that leads to the most visited state (robust child) as follow:

$$i_{\nu_t} = \arg \max_{i_{\nu_t}} \pi_e(i_{\nu_t}|s_t)$$

$$\pi_e(i_{\nu_t}|s_t) = \frac{N(s_t, i_{\nu_t})}{\sum_{b \in \mathcal{C}_u} N(s_t, b)}, \quad (4)$$

where  $\pi_e(s_t) = [\pi_e(i|s_t)]_{i \in \mathcal{C}_{s_u}}^\top$  is the improved policy after MCTS.

### 3.3 Generalizing with Policy and Value Neural Networks

The policy and value neural network is learned to mimic the searching policy  $\pi_e$  and predicts the expected return  $v_{\pi_e}$  of the following  $\pi_e$ . Specifically, the parameters  $\theta$  are adjusted by gradient descent on a loss  $\ell$  over the cross-entropy loss between  $\pi_\theta$  and the search policy  $\pi_e$ , and the mean-squared error between the predicted value  $v_\theta(s_t)$  and the sample discount reward  $z_t$  by following  $\pi_e$  in training stage as:

$$\ell(\theta) = \{(z_t - v_\theta(s_t))^2 - \pi_e(s_t)^\top \log \pi_\theta(s_t) - (1 - \pi_e(s_t))^\top \log(1 - \pi_\theta(s_t))\} + \rho \|\theta\|^2, \quad (5)$$

where  $\pi_\theta(s_t) = [\pi_\theta(i|s_t)]_{i \in \mathcal{C}_{s_u}}^\top$ , and  $\rho$  is a parameter controlling the level of  $\ell_2$  weight regularization.

## 4 Dealing with Large-Scale Datasets

One disadvantage of DIV-MCTS is that the heuristic search over a large candidate item set is time-cost especially when dealing with massive datasets. For example, when the candidate item set contains 1000 items (not a very big item set), they will be nearly 1000 nodes to be evaluated at every expansion, which means that every node has nearly 1000 child nodes. Additionally, the expansion operation will be repeated over and over again until the maximum iteration is reached. Searching over such a broad tree would be problematic. To alleviate this problem, we propose two approaches to solve this problem.

## 4.1 MCTS with Structure Pruning

The first improvement is to prune the branches that seem impossible to be the optimal recommendations before expanding the leaf node. To reduce meaningless search, the structure of optimal recommendations has been incorporated into the expansion of the search tree to efficiently narrow down the searching space.

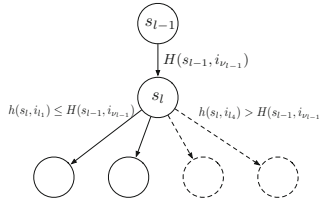
**Lemma 1.** *Assuming the undiscounted accuracy and diversity gain of item  $i$  at  $t$  is  $h_t(i) = 2 \frac{G(i) \times \alpha - G(i)}{G(i) + \alpha - G(i)}$ , and the optimal ranking for  $s_t = \{u, \mathcal{O}_u, \mathbf{i}_{\nu_{<t}}\}$  with  $\mathcal{C}_u$  is  $\mathbf{i}_{\nu_{\geq t}}^* = [i_{\nu_t}^*, i_{\nu_{t+1}}^*, \dots, i_{\nu_N}^*]$ , then we have  $h_t(i_{\nu_t}^*) \geq h_t(i_{\nu_{t+1}}^*) \geq \dots \geq h_t(i_{\nu_N}^*)$ .*

*Proof.*  $\because$  the accuracy and diversity gain is discounted with  $t$ .

$\therefore$  the total reward is proportional to  $\sum_{j=t}^N \frac{h_j(i_{\nu_j}^*)}{\log_2(j+1)}$ .

$\forall t_1, t_2 \in [t, N], t_1 < t_2$ , if  $h_{t_1}(i_{\nu_{t_1}}^*) < h_{t_1}(i_{\nu_{t_2}}^*)$ , then  $[i_{\nu_{t_1}}^*, \dots, i_{\nu_{t_2}}^*, \dots, i_{\nu_{t_1}}^*, \dots, i_{\nu_N}^*]$  would be the optimal ranking, which is conflict with the assumption that  $\mathbf{i}_{\nu_{\geq t}}^*$  is the optimal ranking. So, we have  $h_t(i_{\nu_t}^*) \geq h_t(i_{\nu_{t+1}}^*) \geq \dots \geq h_t(i_{\nu_N}^*)$ .  $\square$

Lemma 1 shows that the optimal recommendations must be descend on the undiscounted accuracy and diversity gains, which is a very valuable property for pruning unpromising leaf node in tree search. Combined with naive MCTS, we stop the expanding of a leaf node if the undiscounted accuracy and diversity gain is bigger than the parent node. Specifically, in FMCTS, the action node in the tree stores  $\{N(s, i), P(s, i), Q(s, i), H(s, i)\}$ , where  $H(s, i)$  is the undiscounted immediate reward. During the expansion, the leaf node is expanded only when the undiscounted reward of edge  $h(s_l, i_{\nu_l})$  is smaller than the parent node's  $H(s_{l-1}, i_{\nu_{l-1}})$ . Finally, the new edge is initialized with  $H(s_l, i_{\nu_l}) = h(s_l, i_{\nu_l})$ . Figure 2 plots the pruning expansion process.



**Fig. 2.** The structure pruning of node-expansion on MCTS.

## 4.2 Acceleration with Problem Decomposition

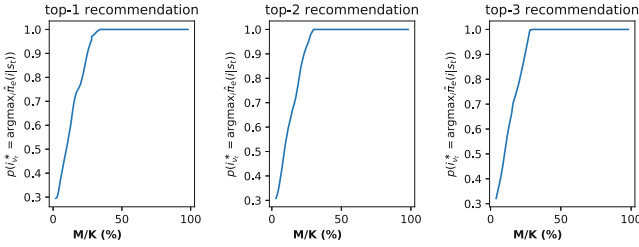
Although the efficiency problem can be alleviated by pruning branches of the search tree, training over such large candidate item set can still be problematic. The intuitive idea is to reduce the size of the candidate item set, which is helpful for harvesting the fruits of tree search more frequently and sharing this knowledge over different tree search. To reduce the search space, we propose to decompose the top-N recommendation problem into many sub-problems

(e.g. searching over a small subset of candidate item set  $\mathcal{C}_u$ ). Then, sharing the searching results by the generalization of the neural network. In Theorem 1, we prove that searching over a randomly chosen subset  $\mathcal{C}_{s_u} \subset \mathcal{C}_u$  will discover the optimal policy in the limits under a mild condition.

**Theorem 1.** *Assuming that the optimal policy for  $s_t = \{u, \mathcal{O}_u, \mathbf{i}_{\nu < t}\}$  under candidate item set  $\mathcal{C}_u$  is  $\pi_e$ . If the searching over a randomly chosen subset  $\mathcal{C}_{s_u} \subset \mathcal{C}_u$  satisfying  $(1 + \frac{M}{K}) \frac{(M-1)!(K-N)!}{(K-1)!(M-N)!} \geq 1$ , where  $M = |\mathcal{C}_{s_u}|$ ,  $K = |\mathcal{C}_u|$ ,  $K, M \gg N$ .  $\forall p, q \in \mathcal{C}_u$  with  $\pi_e(i_{\nu_t}^p | s) \geq \pi_e(i_{\nu_t}^q | s)$ , we have  $\mathbb{E}[\hat{\pi}_e(i_{\nu_t}^p | s)] \geq \mathbb{E}[\hat{\pi}_e(i_{\nu_t}^q | s)]$ , where  $\hat{\pi}_e$  is the optimal planning policy under  $\mathcal{C}_{s_u}$ .*

*Proof.* Assuming that the optimal ranking under  $\mathcal{C}_u$  starts with  $i_{\nu_t}^p$  is  $\mathbf{i}_{\nu_{\geq t}}^p = [i_{\nu_t}^p, i_{\nu_{t+1}}^p, \dots, i_{\nu_N}^p]$ . From Lemma 1, we can infer that  $\mathbb{E}[\hat{\pi}_e(i_{\nu_t}^p | s)] \geq \mathbb{E}[\hat{\pi}_e(i_{\nu_{t+i}}^p | s)]$ ,  $\forall i \geq 1$ .

For a item  $i_{\nu_t}^q$  ( $i_{\nu_t}^q \notin \mathbf{i}_{\nu_{\geq t}}^p$ ), the probability of  $\mathbf{i}_{\nu_{\geq t}}^p \subset \mathcal{C}_{s_u}$  is  $p(\mathbf{i}_{\nu_{\geq t}}^p \subset \mathcal{C}_{s_u}) = \frac{C_{K+t-(N+1)}^{M+t-(N+1)}}{C_K^M}$  and  $p(\mathbf{i}_{\nu_{\geq t}}^p \not\subset \mathcal{C}_{s_u} \text{ and } i_{\nu_t}^q \in \mathcal{C}_{s_u}) = (1 - \frac{C_{K+t-(N+1)}^{M+t-(N+1)}}{C_K^M}) \frac{C_{K-1}^{M-1}}{C_K^M}$ .



**Fig. 3.** The influence of  $\frac{M}{K}$  on the mismatch of  $\hat{\pi}_e$  and  $\pi_e$ .

Then,

$$\begin{aligned}
 & \mathbb{E}[\hat{\pi}_e(i_{\nu_t}^p | s)] \geq \mathbb{E}[\hat{\pi}_e(i_{\nu_t}^q | s)] \\
 & \Leftrightarrow p(\mathbf{i}_{\nu_{\geq t}}^p \subset \mathcal{C}_{s_u}) \geq p(\mathbf{i}_{\nu_{\geq t}}^p \not\subset \mathcal{C}_{s_u} \text{ and } i_{\nu_t}^q \in \mathcal{C}_{s_u}) \\
 & \Leftrightarrow \frac{C_{K+t-(N+1)}^{M+t-(N+1)}}{C_K^M} \geq (1 - \frac{C_{K+t-(N+1)}^{M+t-(N+1)}}{C_K^M}) \frac{C_{K-1}^{M-1}}{C_K^M} \\
 & \Leftrightarrow 1 \leq (1 + \frac{M}{K}) \frac{(M-1)!(K-N)!}{(K-1)!(M-N)!}
 \end{aligned} \tag{6}$$

So,  $\forall p, q \in \mathcal{C}_u$  with  $\pi_e(i_{\nu_t}^p | s) \geq \pi_e(i_{\nu_t}^q | s)$ , we have  $\mathbb{E}[\hat{\pi}_e(i_{\nu_t}^p | s)] \geq \mathbb{E}[\hat{\pi}_e(i_{\nu_t}^q | s)]$ .  $\square$

The condition in Eq. (6) is a very restrictive, since we assume that  $i_{\nu_t}^q$  would be the optimal solution with  $\mathbf{i}_{\nu_{\geq t}}^p \not\subset \mathcal{C}_{s_u}$ . To get a loosen condition, we study the influence of  $\frac{M}{K}$  on  $\mathbb{E}[\hat{\pi}_e]$  by simulating on a synthetic dataset. For the construction of synthetic dataset, we randomly create multi-candidate item sets  $\mathcal{C}_{s_u}$  with

100 items, 20 topics, and 5 ratings, and searching over a randomly chosen subset  $\mathcal{C}_{s_u}$  with a fixed size  $M$ . Then, MCTS is employed to generate expert policy  $\hat{\pi}_e$ . Finally, we study the difference between  $\hat{\pi}_e$  and  $\pi_e$  by comparing the recommendation chosen by  $\pi_e$  and  $\hat{\pi}_e$ , because it determines the policies' performances. The simulation code is available on <https://github.com/zoulixin93/FMCTS>. In Fig. 3, we show the difference between  $\hat{\pi}_e$  and  $\pi_e$  by comparing the recommendation made by  $\hat{\pi}_e$  and  $\pi_e$ . As we can see, the bigger  $\frac{M}{K}$  means greater possibility of choosing optimal recommendation  $i_{\nu_t}^*$  with  $\hat{\pi}_e$ . For  $M/K \geq 0.25$ , the recommendation made by  $\pi_e$  and  $\hat{\pi}_e$  would be same, which is much more loosen than the condition in Eq. (6). Combining these two techniques with the training framework, the resulting detailed training procedure is provided in Algorithm 1.

## 5 Related Work

In this section, we briefly review works related to our study. In general, the related work can be mainly grouped into the following three categories: recommendation diversity, deep learning based methods, and reinforcement learning based methods.

---

**ALGORITHM 1.** Training of Div-FMCTS

---

**Input:** Training dataset  $\mathcal{D} = \{(u^{(j)}, \mathcal{O}_u^{(j)}, \mathcal{C}_u^{(j)})\}_{j=1}^D$ , learning rate  $\delta$ , maximum iteration number  $\Lambda$ , trade-off parameter  $c_{puct}$ , reward function  $r$ , subset size  $M$ , buffer size  $B$ .  
**Output:** Well-trained  $\pi_\theta$ .

- 1 Randomly initialize parameters  $\theta \leftarrow \mathcal{U}(-0.1, 0.1)$
- 2 Initialize a queue  $\Psi$  to capacity  $B$
- 3 **repeat**
- 4     **for**  $\{(u, \mathcal{O}_u, \mathcal{C}_u)\} \in \mathcal{D}$  **do**
- 5         Randomly chose  $M$  items from  $\mathcal{C}_u$  as the candidate set  $\mathcal{C}_{s_u}$
- 6         **for**  $t$  from 1 to  $N$  **do**
- 7             Set  $s_t \leftarrow \{u, \mathcal{O}_u, i_{\nu < t}\}$
- 8             Do MCTS under  $s_t$  until the maximum iteration number  $\Lambda$  is reached
- 9             Choose  $i_{\nu_t} = \arg \max_{i \in \mathcal{C}_{s_u}} \pi_e(i|s_t)$
- 10             Calculate the reward  $r_t$  with Equation (2)
- 11         **end**
- 12          $z = 0$
- 13         **for**  $t$  from  $N$  to 1 **do**
- 14              $z \leftarrow r_t + \gamma z$
- 15             Store the tuple  $\{u, \mathcal{O}_u, i_{\nu < t}, z, \pi_e\}$  in  $\Psi$
- 16         **end**
- 17         Sample a mini-batch of  $\{u, \mathcal{O}_u, i_{\nu < t}, z, \pi_e\}$  from  $\Psi$
- 18         Update parameters  $\theta \leftarrow \theta - \delta \frac{\partial \ell(\theta)}{\partial \theta}$  with Equation (5)
- 19     **end**
- 20 **until** converge;

---

### 5.1 Recommendation Diversity

Recommendation diversity refers to aggregate diversity and individual diversity. **Aggregate diversity** means the overall diversity of all users and is being measured by using absolute and relative long-tail metrics [1, 4]. Adomavicius et al. [1] firstly proposed a graph-theoretic approach for maximizing aggregate

recommendation diversity based on maximum flow or maximum bipartite matching computations. **Individual diversity** aims at maximizing the recommendation diversity for each user and is usually being measured by topic coverage or  $\alpha$ -*NDCG* [19,41]. In this study, the focus is on individual diversity.

The methods for individual diversity improvement can be divided into two categories. The first category is the point-wise methods, which involves two phases: generating the accuracy estimates of items and heuristically ranking the items considering both the accuracy and diversity. For example, a popular heuristic re-ranking approach has been proposed in [2]. Ziegler et al. [41] came up with a novel approach named topic diversification to balance and diversify personalized recommendations. Azin et al. [3] proposed a greedy selection method for diversifying individual recommendation. The second category is the list-wise approach, which directly generates a subset accurate and diverse recommendation. For example, Cheng et al. [5] presented a supervised learning method, which combined parameterized matrix factorization and structural learning to seek significant improvement in diversity while maintaining accuracy. However, point-wise methods require fine-tuning features for heuristic methods. Additionally, leveraging heuristic method possibly only find the suboptimal solution to the problem. For list-wise approach, ground truth labels for training is needed, which is usually obtained by heuristic methods and can not scale for the large datasets.

## 5.2 Deep Learning Based Recommendation

With the booming success of deep learning in computer vision and natural language processing [15], there are many works trying to apply neural networks for recommender systems. The first work is proposed in [27], which uses a variant of the restricted boltzmann machine for collaborative filtering (RBM-CF). Similar to RBM, Autoencoders and the stacked-denoising autoencoders based recommender systems have been analyzed in [18,31], which are compact and efficiently trainable model. Recently, Zheng et al. [40] introduced a neural autoregressive approach for collaborative filtering (CF-NADE), which has beat the state-of-the-art methods on MovieLens and Netflix datasets. Xue et al. [12] proposed to use DNN to learn the latent factors from the sparse user-item matrix and make recommendations based on the relevance measure of user-item latent factors. There are also many hybrid methods, which combine matrix factorization (MF) with DNN to solve the problem of cold start, such as [34,35]. Apart from the traditional recommender, recurrent neural networks have been widely employed to sequential recommender system, such as GRU4Rec [10], NARM [17]. In spite of its success, all these methods are mainly concerned with increasing the recommendation accuracy in traditional user-item setting and none of these works could directly optimize the diversity of recommendation.

### 5.3 Reinforcement Learning Based Methods

Since the great success in playing Atari 2600 video games and Go at a superhuman level directly from image pixels [23, 24, 29], deep reinforcement learning has attracted enormous research interests. However, due to the difficulty of dealing with larger action space, there are just a few works applying the techniques of deep reinforcement learning to the recommender system. For example, in [28], an MDP based recommender system has been proposed, which recommends most likely purchasing items by using a sample-based transition distribution. It requires huge samples to learn the transition distribution and is limited to solve the problem with small state space. To maximize the accumulative rewards, Mahmood et al. [21] adopted the reinforcement learning technique to optimize the responses of users in a conversational recommender. Sunehag et al. [32] introduced slate-MDPs and successfully addressed sequential decision problems with high-dimensional combinatorial slate-action spaces. Arnold et al. [7] proposed an actor-critic based model to deal with the problem of large discrete action spaces. It leverages the policy gradient to learn a policy mapping from state to a continuous action and chooses the action with the biggest  $Q$ -value in candidate actions, which is generated by finding the  $k$ -nearest neighbors of continuous action in discrete action space. In [20], it proposed to use the latent factors learned from probability matrix factorization (PMF) as the belief states and approximate the  $Q$ -value by using a neural network with the latent factors as input.

Apart from the recommendation scene, reinforcement learning has also been applied to information retrieval. For example, in [38], MDP has been adapted to solve the search result diversification by using policy gradient. Yisong et al. formulated [39] the real-time learning in the search engine as a dueling bandit.

## 6 Experiments

In this section, we conduct experiments to demonstrate the effectiveness of our proposed method. We also do some extensive experiments to analysis the effectiveness of structure pruning and the decomposition of diversity recommendation.

### 6.1 Experimental Settings

*Dataset.* Since we mainly focus on the recommendation diversity, the chosen datasets have to satisfy a fundamental requirement: each item should be associated with topics for generating training instances and evaluating the predictions. In this study, fours benchmark datasets MovieLens 100k, 1M, 10M and 20M<sup>4</sup> have been chosen to evaluate Div-FMCTS. The statistics of datasets are shown in Table 1.

---

<sup>4</sup> <https://grouplens.org/datasets/movielens/>.



**Table 1.** Summary statistics of datasets.

Dataset	#user	#item	#rating	#topic	average topics
MovieLens100k	943	1,682	100,000	19	1.72
MovieLens1M	6,040	3,900	1,000,209	19	1.67
MovieLens10M	71,567	10,681	10,000,054	18	2.02
MovieLens20M	138,493	26,774	20,000,263	20	2.00

*Baselines.* First of all, we choose DCF [5] as our main baseline, which is a strong baseline on diverse top-N recommendation. Besides, since Div-FMCTS is a reinforcement learning based algorithm, we also choose REINFORCE [36] as our baseline. Additionally, we also compare Div-FMCTS with a RNN-based recommendation model NARM [17] and the most popular and widely used matrix factorization (MF) [16].

*Evaluation.* Like most diverse recommender systems [5], we choose  $NDCG@N$  and  $\alpha\text{-}NDCG@N$  as the metrics to evaluate the effectiveness on recommendation accuracy and diversity. Furthermore, we also utilize the  $F$ -measure, named  $F_{NDCG@N}$ , to assess the trade-off performance with consideration of both accuracy and diversity.

*Parameter Settings.* For each datasets, 10% of the ratings are randomly selected as the test set, leaving the remaining 90% of the ratings as the training set. Among the ratings in the training set, 5% are used as validation set, which is used to find the optimal hyperparameters. For neural network, we randomly initialized model parameters with a uniform distribution ranging from  $-0.01$  to  $0.01$ , optimizing the model with mini-batch Adam [14]. In MCTS, the trade-off parameter for exploitation and exploration is set as  $c_{puct} = 2000$ . The  $\frac{M}{K}$  is set as  $0.2$  for  $K \geq 100$ . The maximum iteration number of MCTS is  $1000$  on experiments. Following the setting of DCF [5], we mainly consider top-3 recommendation problem, where a list of 3 items is recommended for each user. The models are defined and trained in Tensorflow on a Nvidia Tesla P40 GPU. The source code for Div-FMCTS can be found at Github: <https://github.com/zoulixin93/FMCTS>.

## 6.2 Experimental Results

*Comparison Against Baselines.* We compared Div-FMCTS with state-of-the-art methods. The results of all methods over the benchmark datasets in terms of three metrics are shown in Table 2. From the results we can see that, Div-FMCTS outperformed all of the baseline methods on  $\alpha\text{-}NDCG@3$  and  $F_{NDCG@3}$  by a large margin. And also, even if it is not the best result, it is not much different from the best result in term of  $\alpha\text{-}NDCG@3$ . We conducted significance testing ( $t$ -test) on the improvements of our approaches over the all baselines.  $\blacktriangledown$  denotes

strong significant divergence with  $p\text{-value} < 0.05$ . The improvements are significant, in terms of  $\alpha\text{-}NDCG@3$  and  $F_{NDCG}@3$ . The results indicate that our model Div-FMCTS can indeed improve recommendation diversity while maintaining recommendation accuracy.

**Table 2.** Comparison between Div-FMCTS and state-of-the-art methods.

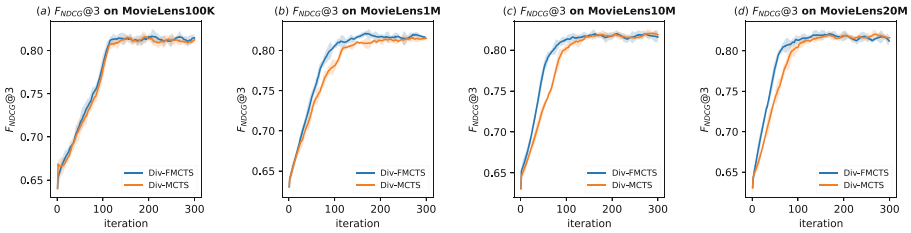
Methods	MovieLens100k			MovieLens1M		
	$NDCG@3$	$\alpha\text{-}NDCG@3$	$F_{NDCG}@3$	$NDCG@3$	$\alpha\text{-}NDCG@3$	$F_{NDCG}@3$
Div-FMCTS	0.8080	<b>0.8364</b>	<b>0.8222</b>	0.8288	<b>0.8277</b>	<b>0.8282</b>
REINFORCE	0.7523 $\blacktriangledown$	0.6291 $\blacktriangledown$	0.6888 $\blacktriangledown$	0.7546 $\blacktriangledown$	0.6075 $\blacktriangledown$	0.6731 $\blacktriangledown$
DCF <sup>a</sup>	0.7070	0.7414	0.7238	0.7415	0.6735	0.7059
NARM	<b>0.8213</b>	0.5208 $\blacktriangledown$	0.6374 $\blacktriangledown$	<b>0.8305</b>	0.4935 $\blacktriangledown$	0.6191 $\blacktriangledown$
MF	0.8041 $\blacktriangledown$	0.5346 $\blacktriangledown$	0.6422 $\blacktriangledown$	0.8023 $\blacktriangledown$	0.5011 $\blacktriangledown$	0.6169 $\blacktriangledown$
	MovieLens10M			MovieLens20M		
Div-FMCTS	<b>0.8274</b>	<b>0.8201</b>	<b>0.8237</b>	0.8053	<b>0.8278</b>	<b>0.8165</b>
REINFORCE	0.7277 $\blacktriangledown$	0.6442 $\blacktriangledown$	0.6834 $\blacktriangledown$	0.7325 $\blacktriangledown$	0.6232 $\blacktriangledown$	0.6734 $\blacktriangledown$
NARM	0.8206	0.5098 $\blacktriangledown$	0.6289 $\blacktriangledown$	<b>0.8310</b>	0.4782 $\blacktriangledown$	0.6071 $\blacktriangledown$
MF	0.8047 $\blacktriangledown$	0.4774 $\blacktriangledown$	0.5992 $\blacktriangledown$	0.8179 $\blacktriangledown$	0.4702 $\blacktriangledown$	0.5971 $\blacktriangledown$

<sup>a</sup>The experimental results of DCF is taken from [5].

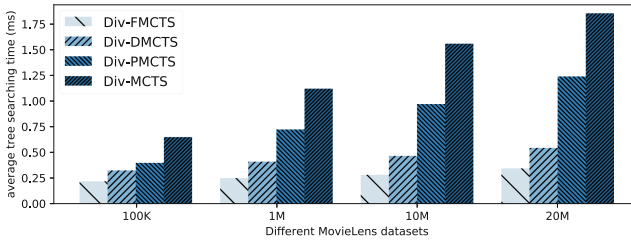
*Comparison Between FMCTS and Naive MCTS.* It is a key question that how much impact on the learned policy  $\hat{\pi}_\theta$  through MCTS over a subset of candidate item set. To investigate its influence, we train a policy  $\hat{\pi}_\theta$  by MCTS over the complete candidate set  $\mathcal{C}_u$ , named Div-MCTS. For the sake of fairness, Div-FMCTS and Div-MCTS are trained with same hyperparameters. In Fig. 4, we show the  $F_{NDCG}@3$  w.r.t. the training iteration. The blue one is the performance of training with problem decomposition and the orange one is training with the complete candidate item set  $\mathcal{C}_u$ . From Fig. 4, the  $F_{NDCG}@3$  is nearly same on these four benchmark dataset, which means that training over a subset of candidate set does not influence the performance. Meanwhile, Div-FMCTS converges faster than Div-MCTS, which indicates that MCTS over a subset of candidate item set can do the optimization more efficiently.

*The Advantage of FMCTS on Searching Time.* To examine the advantage of Div-MCTS, we compared average searching time of four different MCTS: **(1)** Div-MCTS is searching over the whole candidate item set without structure pruning; **(2)** Pruning the MCTS with the descend immediate reward – Div-PMCTS; **(3)** MCTS with problem decomposition, named Div-DMCTS; **(4)** The proposed Div-FMCTS, searching with a smaller candidate set and structure pruning. In our experiments, the tree searches are running on a server with 64 GB memory and an INTEL Core i7-7800X CPU, the maximum iteration number of MCTS are 1000. Figure 5 plots the average searching time over these four datasets. We can see that the searching time of Div-FMCTS and Div-DMCTS do not increase too much with respect to the increasing number of items. In contrast, the average searching time of Div-MCTS and Div-PMCTS grow fast w.r.t the increasing size of datasets. Additionally, using structure pruning will decrease planning time.

From the results, we can see that searching over a subset of candidate item set and pruning branches with descend immediate reward is helpful on saving the training time.



**Fig. 4.** The influence of planning over a subset of candidate item set.



**Fig. 5.** The average tree searching time of four different MCTS.

## 7 Conclusion

In this paper, we have proposed a novel MDP based method – Div-FMCTS – to directly maximize the trade-off between accuracy and diversity for the top-N recommendation. The learning of Div-FMCTS is decomposed into iteration between searching with MCTS and generalizing those plans with a policy-value neural network. To faster the tree search process, a novel structure pruning technique has been incorporated into the node expansion to narrow down searching space for MCTS. Additionally, we theoretically and empirically verify that the diversity recommendation can be equivalently solved by planning under the sub-problems. Extensive experiments on four benchmark datasets have demonstrated the effectiveness of our proposed method.

## References

1. Adomavicius, G., Kwon, Y.: Maximizing aggregate recommendation diversity: a graph-theoretic approach. In: RecSys, pp. 3–10 (2011)
2. Adomavicius, G., Kwon, Y.: Improving aggregate recommendation diversity using ranking-based techniques. TKDE **24**(5), 896–911 (2012)
3. Ashkan, A., Kveton, B., Berkovsky, S., Wen, Z.: Optimal greedy diversity for recommendation. In: IJCAI, pp. 173–182 (2015)

4. Brynjolfsson, E., Hu, Y., Smith, M.D.: Research commentary-long tails vs. superstars: the effect of information technology on product variety and sales concentration patterns. *Inf. Syst. Res.* **21**(4), 736–747 (2010)
5. Cheng, P., Wang, S., Ma, J., Sun, J., Xiong, H.: Learning to recommend accurate and diverse items. In: *WWW*, pp. 183–192 (2017)
6. Clarke, C.L.A., et al.: Novelty and diversity in information retrieval evaluation. In: *SIGIR*, pp. 659–666 (2008)
7. Dulac-Arnold, G.: Deep reinforcement learning in large discrete action spaces. arXiv preprint [arXiv:1512.07679](https://arxiv.org/abs/1512.07679) (2015)
8. Feige, U.: A threshold of  $\ln n$  for approximating set cover. *JACM* **45**(4), 634–652 (1998)
9. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.-S.: Neural collaborative filtering. In: *WWW*, pp. 173–182 (2017)
10. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based recommendations with recurrent neural networks. arXiv (2015)
11. Hochba, D.S.: Approximation algorithms for NP-hard problems. *ACM SIGACT News* **28**(2), 40–52 (1997)
12. Xue, H.-J., Dai, X.-Y., Zhang, J., Huang, S., Chen, J.: Deep matrix factorization models for recommender systems. In: *IJCAI*, pp. 764–773 (2017)
13. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst. (TOIS)* **20**(4), 422–446 (2002)
14. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980) (2014)
15. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. *Nature* **521**(7553), 436–444 (2015)
16. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: *NIPS 2001*, pp. 556–562 (2001)
17. Li, J., Ren, P., Chen, Z., Ren, Z., Ma, J.: Neural attentive session-based recommendation. arXiv preprint [arXiv:1511.06939](https://arxiv.org/abs/1511.06939) (2015)
18. Li, S., Kawale, J., Fu, Y.: Deep collaborative filtering via marginalized denoising auto-encoder. In: *CIKM*, pp. 811–820 (2015)
19. Liu, T.-Y., et al.: Learning to rank for information retrieval. *Found. Trends® Inf. Retr.* **3**(3), 225–331 (2009)
20. Lu, Z., Yang, Q.: Partially observable Markov decision process for recommender systems. arXiv preprint [arXiv:1608.07793](https://arxiv.org/abs/1608.07793) (2016)
21. Mahmood, T., Ricci, F.: Improving recommender systems with adaptive conversational strategies. In: *HT 2009*, pp. 73–82 (2009)
22. McNee, S.M., Riedl, J., Konstan, J.A.: Being accurate is not enough: how accuracy metrics have hurt recommender systems. In: *CHI*, pp. 1097–1101 (2013)
23. Mnih, V., et al.: Playing atari with deep reinforcement learning. arXiv preprint [arXiv:1312.5602](https://arxiv.org/abs/1312.5602) (2013)
24. Mnih, V., et al.: Human-level control through deep reinforcement learning. *Nature* **518**(7540), 529–533 (2015)
25. Rendle, S.: Factorization machines. In: *ICDM*, pp. 995–1000 (2007)
26. Salakhutdinov, R., Mnih, A.: Probabilistic matrix factorization. In: *NIPS*, pp. 1257–1264 (2007)
27. Salakhutdinov, R., Mnih, A., Hinton, G.: Restricted Boltzmann machines for collaborative filtering. In: *ICML*, pp. 791–798 (2007)
28. Shani, G., Heckerman, D., Brafman, R.I.: An MDP-based recommender system. *JMLR* **6**, 1265–1295 (2005)

29. Silver, D., et al.: Mastering the game of go with deep neural networks and tree search. *Nature* **529**(7587), 484–489 (2016)
30. Silver, D., et al.: Mastering the game of go without human knowledge. *Nature* **550**, 354–360 (2017)
31. Strub, F., Mary, J.: Collaborative filtering with stacked denoising autoencoders and sparse inputs. In: *NIPS*, pp. 111–112 (2015)
32. Sunehag, P., Evans, R., Dulac-Arnold, G., Zwols, Y., Visentin, D., Coppin, B.: Deep reinforcement learning with attention for slate Markov decision processes with high-dimensional states and actions. *arXiv preprint [arXiv:1512.01124](https://arxiv.org/abs/1512.01124)* (2015)
33. Szpektor, I., Maarek, Y., Pelleg, D.: When relevance is not enough: promoting diversity and freshness in personalized question recommendation. In: *WWW*, pp. 173–182 (2013)
34. den Oord, A.V., Dieleman, S., Schrauwen, B.: Deep content-based music recommendation. In: *NIPS*, pp. 2643–2651 (2015)
35. Wang, H., Wang, N., Yeung, D.-Y.: Collaborative deep learning for recommender systems. In: *SIGKDD*, pp. 1235–1244 (2015)
36. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* **8**(3–4), 229–256 (1992)
37. Wu, Y., DuBois, C., Zheng, A.X., Ester, M.: Collaborative denoising auto-encoders for top-n recommender systems. In: *WSDM*, pp. 153–162 (2016)
38. Xia, L., Xu, J., Lan, Y., Guo, J., Zeng, W., Cheng, X.: Adapting Markov decision process for search result diversification. In: *SIGIR*, pp. 535–544 (2017)
39. Yue, Y., Joachims, T.: Interactively optimizing information retrieval systems as a dueling bandits problem. In: *ICML*, pp. 1201–1208 (2009)
40. Zheng, Y., Tang, B., Ding, W., Zhou, H.: A neural autoregressive approach to collaborative filtering. In: *ICML*, pp. 764–773 (2016)
41. Ziegler, C.-N., McNee, S.M., Konstan, J.A., Lausen, G.: Improving recommendation lists through topic diversification. In: *WWW*, pp. 22–32 (2005)



# Retweeting Prediction Using Matrix Factorization with Binomial Distribution and Contextual Information

Bo Jiang<sup>1</sup>, Zhigang Lu<sup>1,2</sup>, Ning Li<sup>1(✉)</sup>, Jianjun Wu<sup>1,2</sup>, Feng Yi<sup>3</sup>,  
and Dongxu Han<sup>1,2</sup>

<sup>1</sup> Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China  
{jiangbo,luzhigang,lining6,wujianjun,handongxu}@iie.ac.cn

<sup>2</sup> School of Cyber Security, University of Chinese Academy of Sciences,  
Beijing, China

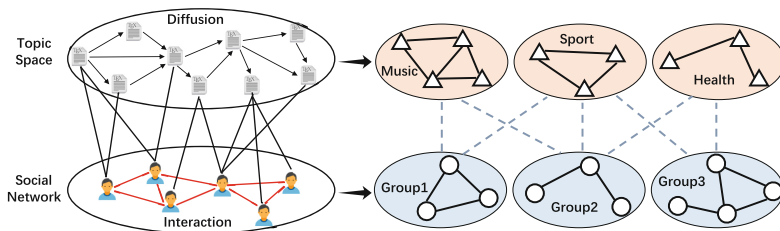
<sup>3</sup> School of Computer Engineering,  
University of Electronic Science and Technology of China, Zhongshan, China  
bjstarbow@gmail.com

**Abstract.** Retweeting provides an efficient way to expand information diffusion in social networks, and many methods have been proposed to model user's retweeting behaviors. However, most of existing works focus on devising an effective prediction method based on social network data, and few research studies explore the data characteristic of retweeting behaviors which is typical binary discrete distribution and sparse data. To this end, we propose two novel retweeting prediction models, named Binomial Retweet Matrix Factorization (BRMF) and Context-aware Binomial Retweet Matrix Factorization (CBRMF). The two proposed models assume that retweetings are from binomial distributions instead of normal distributions given the factor vectors of users and messages, and then predicts the unobserved retweetings under matrix factorization. To alleviate data sparsity and reduce noisy information, CBRMF first learns user community by using community detection method and message clustering by using short texts clustering algorithm from social contextual information on the basis of homophily assumption, respectively. Then CBRMF incorporates the impacts of homophily characteristics on users and messages as two regularization terms into BRMF to improve the prediction performance. We evaluate the proposed methods on two real-world social network datasets. The experimental results show BRMF achieves better the prediction accuracy than normal distributions based matrix factorization model, and CBRMF outperforms existing state-of-the-art comparison methods.

**Keywords:** Retweeting prediction · Social network · Binomial distribution · Contextual information · Matrix factorization

# 1 Introduction

Recent years have witnessed an increasing amount of social network services such as Twitter, Facebook and Weibo. One of the distinguishing features of these services is the retweeting mechanism which forwards messages published by other users and shares them with one’s own followers. Most of existing studies have shown that retweeting is considered as a key mechanism of information diffusion in social networks [1, 18]. Taking full advantage of the function, one can achieve better insights into the process of information diffusion, making a right strategic decision for many tasks of social network applications such as event discovery [10], community detection [3, 28], and recommender system [5, 16]. Thus, understanding the influencing factors of retweetings from the observed data and predicting the hidden mechanism underlying diffusion is a critical and fundamental task in these applications.



**Fig. 1.** An example of social groups on social network. Users have diverse topical interests, each group corresponding to a user community with common interests. Messages also have the diversity topics, each topic corresponding to a topic space with similar semantic content.

Considerable work has been carried out on investigating the influencing factors of retweeting decision through user survey [1, 18] and statistical analysis [23, 30]. These studies find that the interests of user’s topics and the strength of social influence are two most important influencing factors when user decides to retweet a message. Compared with the above efforts, more and more studies have been getting to focus on devising an effective retweeting prediction model from different perspectives. For example, a simple but powerful strategy is to consider the retweeting prediction as a classification problem by extracting the different features like user’s profile, network structure and message’s content, historical interactions [2, 9, 12]. Although these studies can solve the problem of retweeting prediction to some extent, there is lack of a more principled way to extract the set of related features. The social influence-based models are also proposed to quantify the strength of user influence from the views of network structure and historical interactions [11, 32]. However, these models depend on the different types of information, which may not be always available in some platforms. The factor graph-based methods are used to represent factorization of retweeting probability distribution function [19, 33]. However, such models are

too complex and difficult to be applicable to large-scale applications. Recently, the attention-based deep neural network method is proposed by incorporating social contextual information for this task [34]. However, the training process of neural networks requires a large amount of labeled data, it is not always available in most social networks. Another matrix factorization-based methods convert the problem into matrix completion based on the observed entities [25, 26]. These methods leverage the power of matrix factorization model to achieve a relatively high accuracy of prediction by incorporating explicit information and implicit feedback. However, none of methods focus on the data characteristic of retweeting behaviors with binary distributions on social networks.

The present findings demonstrate that social users and information flows naturally form social communities and topic clusters underlying network structure and interactions, as shown in Fig. 1. Moreover, many social networks such as Facebook and Google+ provide “social group” function, which allows the users to be incorporated into a new densely connected subgraph with the same or similar personal interest preference. In this case, user’s decision is strongly influenced by the other active neighbors from his friends [32]. On the other hand, information flows can also be mapped into the topic space where clusters of messages form topics. Therefore, we argue that the problem of retweeting prediction should be modeled by considering discrete distribution of social data and the impacts of homophily characteristics on users and messages.

In this paper, we propose two novel matrix factorization methods for the retweeting prediction, named Binomial Retweet Matrix Factorization (BRMF) and Context-aware Binomial Retweet Matrix Factorization (CBRMF). The two models assume retweetings are from binomial distributions instead of normal distributions. CBRMF is an extended BRMF model by considering social circles and message clustering to alleviate the data sparsity and reduce the noisy information based on the impacts of homophily.

**Contributions.** In this paper, we make the following three contributions:

- We propose two novel matrix factorization methods for retweeting prediction problem. The two methods assume retweetings are from binomial distributions instead of normal distributions. To the best of our knowledge, this is the first work for retweeting prediction to exploit binomial distributions with the matrix factorization model.
- We utilize user community and document clustering as contextual regularization terms into binomial retweet matrix factorization based on homophily assumption to alleviate the data sparsity and reduce the noisy information, as well as improve the performance of prediction.
- We conduct several analysis experiments with two real-world social network datasets, the experimental results demonstrate our proposed models outperform state-of-the art comparison methods.

**Outline.** The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 provides the formal definition of the retweeting prediction problem and introduces notations used in this paper. In Sect. 4, we present two



retweeting prediction model based on binomial distributions and its learning and inference procedures. Section 5 evaluates our proposed models with the real large social network datasets in terms of accuracy. Section 6 gives a conclusion of this work.

## 2 Related Work

**Social Recommendation with Matrix Factorization.** Recommender systems are used as an efficient tool for dealing with the information overload problem. Many social recommendation approaches have been developed in recent years. For example, SoRec [13] employs both users' social network information and rating records to solve the data sparsity and poor prediction accuracy problems based on probabilistic matrix factorization. Similarly, Context MF [8] incorporates individual preference and interpersonal influence based on probabilistic matrix factorization for improving the accuracy of social recommendation. Ensemble methods predict a missing rating for a given user by a linear combination of ratings from the user and the social network. STE [14] represents the formulation of the social trust restrictions by fusing the users' tastes and their trusted friends' favors together on the recommender systems. mTrust [24] studies multi-faceted trust relationships between users for rating prediction. Regularization methods focus on a user's preference and assume that a user's preference should be similar to that of her social network. SocialMF [6] incorporates the mechanism of trust propagation into the matrix factorization approach for recommendation in social networks. Social Regularization [15] imposes social regularization terms to constrain matrix factorization objective functions based on users' social friend information. TBPR [27] studies the effects of distinguishing strong and weak ties by using neighbourhood overlap to approximate tie strength in social recommendation. In summary, social recommendation methods have been successfully applied in the missing values prediction tasks.

**Retweet Behavior Modeling.** Existing studies focus on exploring the influencing factors of retweeting behaviors by conducting user survey [1, 18] and performing empirical analysis [17, 22, 29]. These results indicate that the intention of user retweeting message is positively influenced by sharing the informative content of message, and enhancing social influence from social relationships. On the other hand, more efforts have been guided on modeling how a message be retweeted in social network. For instance, feature-based methods take the set of related features to predict retweeting behavior such as social features [7, 12], visual features [4]. Social influence-based methods also have been proposed to model user retweeting behavior [32]. Most of the above methods apply heuristic methods to extract the set of features for retweeting prediction. However, some of these features may be computationally expensive or not always available in some social networks. Zhang et al. [33] propose a novel nonparametric Bayesian model adapted from the hierarchical Dirichlet process to combine textual, structural, and temporal information for the task. Subsequently, Zhang et al. [34]

also propose a attention-based deep neural network retweet model by incorporating the user, author, user interests, and similarity information between the tweets and user interests. Besides, other studies employ matrix factorization by using social contextual information from user and content dimensions to solve the problem [26]. Nevertheless, no research considers discrete distributions of retweeting values on social networks. The retweeting prediction has still some unsolved problems such as exploring the data characteristic and studying the role of group structure on users and messages.

### 3 Problem Preliminaries

We give some necessary notations used in this paper and present a formal representation of the retweeting prediction problem under the probabilistic matrix factorization model.

	A	B	C	D
Alice	1	0	?	0
Bob	0	1	0	?
Carl	?	1	0	1
David	1	0	?	0
Eric	0	?	1	0

(a) An example of a binary data with unknowns. Users are presented in rows, while messages are in columns.

$$R = \begin{bmatrix} 1 & 0 & ? & 0 \\ 0 & 1 & 0 & ? \\ ? & 1 & 0 & 1 \\ 1 & 0 & ? & 0 \\ 0 & ? & 1 & 0 \end{bmatrix}$$

(b) Retweeting data can be represented in the matrix  $R$ , which is usually sparse with a high percentage of missing values.

**Fig. 2.** Matrix representation with retweeting data.

Given  $M$  users and  $N$  messages, the behaviors of users retweeting messages are represented in an  $M \times N$  retweeting matrix  $R = [\mathbf{r}_1, \dots, \mathbf{r}_N]$ , in which each row corresponds to a user and each column corresponds to a message. Retweeting has only two states, where  $R_{ij}$  takes the value of 1 if  $u_i$  retweets  $m_j$  and 0 otherwise. Let  $U \in \mathbb{R}^{K \times M}$  and  $V \in \mathbb{R}^{K \times N}$  be the latent user and message feature matrices respectively, where  $U_i$  represents a user and  $V_j$  represents a message in latent feature space.  $K$  is the number of the latent features. The likelihood function of the observed retweetings is factorized across  $M$  users and  $N$  messages with each factor based on Probabilistic Matrix Factorization (PMF) [20] as

$$P(R|U, V, \sigma_R^2) = \prod_{i=1}^M \prod_{j=1}^N [\mathcal{N}(R_{ij}|U_i^T V_j, \sigma_R^2)] \quad (1)$$

where  $\mathcal{N}(\cdot|\mu, \sigma^2)$  is the probability density function of the Gaussian distribution with mean  $\mu$  and variance  $\sigma^2$ .

PMF is learned by maximizing posterior probability of matrices  $U$  and  $V$ , which is equivalent to minimizing sum-of-squares of factorization error as

$$\min_{U,V} \sum_{i=1}^M \sum_{j=1}^N I_{ij} (R_{ij} - U_i^T V_j)^2 + \lambda (\|U\|_F^2 + \|V\|_F^2) \quad (2)$$

where the prior distributions over  $U$  and  $V$  are assumed to zero-mean Gaussian,  $(\|U\|_F^2 + \|V\|_F^2)$  can prevent overfitting,  $\|\cdot\|_F$  denotes the Frobenius norm of the matrix.

It's known that PMF assumes that all ratings are from normal distributions given the corresponding user and item factor vectors. However, the normal assumption in the retweeting prediction problem is not suitable since the values of retweetings only have 0s and 1s, as shown in Fig. 2. For this reason, we will explore how to use binomial distributions to solve the problem in the following section.

## 4 The Proposed Model

With the assumption that retweetings are from binomial distributions, we propose Binomial Retweet Matrix Factorization (BRMF) and Context-aware Binomial Retweet Matrix Factorization (CBRMF) for the retweeting prediction task. CBRMF is an extended BRMF model by incorporating user community and document clustering as social contextual regularization terms to alleviate the data sparsity and reduce the noisy information and improve the performance of prediction. The detailed descriptions of our proposed models are as follows.

### 4.1 Binomial Retweet Matrix Factorization

Since retweeting has only two states, i.e., {retweet, not retweet} in our datasets, the assumption of normal distributions sampled from retweet data doesn't draw the retweeting behaviors. Instead, we assume that all retweetings are from binomial distributions with different preference parameters.

The binomial distributions for retweetings satisfy the following conditions: (1) retweetings are independent and identically distributed, (2) retweeting has only two choices, and (3) the retweeting probability of each user is approximately equal to the ratio of the occurring time of retweetings and the total number of retweetings in our datasets. Here, we replace the Gaussian distribution with the Bernoulli distribution in Eq. (1) as

$$P(R|U, V) = \prod_{i=1}^M \prod_{j=1}^N \mathcal{B}(R_{ij} | \mathcal{S}, U_i^T V_j) \quad (3)$$

where  $\mathcal{B}(k|n, p)$  is the binomial probability mass function with parameters  $n$  and  $p$ , and  $\mathcal{S}$  is the number of retweetings in our datasets. For given a user  $u_i$  and a message  $m_j$ , our goal is to maximize the probability of  $u_i$  retweets  $m_j$  as

$$P(R|U, V) = \prod_{i=1}^M \prod_{j=1}^N p(R_{ij})^{I_{ij}} (1 - p(R_{ij}))^{1-I_{ij}} \quad (4)$$

where  $p(\cdot)$  is the probability that  $u_i$  retweets  $m_j$ .  $I \in \{0, 1\}$  is an indicator matrix where  $I_{ij}$  is equal to 1 if  $u_i$  retweets  $m_j$  and 0 otherwise. The sigmoid function  $g(x) = 1/(1 + \exp(-x))$  bound the range of  $U_i^T V_j$  denoting the user-message association.

To learn the model, we maximize the following likelihood objective function as

$$P(R|U, V) = \prod_{i=1}^M \prod_{j=1}^N (g(R_{ij})^{I_{ij}} (1 - g(R_{ij}))^{1-I_{ij}}) \quad (5)$$

The log of posterior distribution with Eq. (5) is given by

$$\begin{aligned} \mathcal{L}(U, V|R) = & \sum_{i=1}^M \sum_{j=1}^N [I_{ij} \ln \frac{g(R_{ij})}{1 - g(R_{ij})} + \ln(1 - g(R_{ij}))] \\ & - \frac{1}{2\sigma_U^2} \sum_{i=1}^M U_i^T U_i - \frac{1}{2\sigma_V^2} \sum_{j=1}^N V_j^T V_j + Const \end{aligned} \quad (6)$$

where users and messages draw zero-mean Gaussian distributions.

Maximizing Eq. (6) is equivalent to minimizing the following objective function as

$$\sum_{i=1}^M \sum_{j=1}^N [\ln(e^{U_i^T V_j} + 1) - U_i^T V_j I_{ij}] + \frac{\lambda}{2} (\sum_{i=1}^M \|U_i\|_F^2 + \sum_{j=1}^N \|V_j\|_F^2) \quad (7)$$

where  $(\|U_i\|_F^2 + \|V_j\|_F^2)$  are also to avoid overfitting.

The local minimum of the objective function given by Eq. (7) can be found by performing stochastic gradient descent (SGD) approach on feature vectors  $U_i$  and  $V_j$  as

$$\frac{\partial \mathcal{L}}{\partial U_i} = \sum_{j=1}^N \frac{e^{U_i^T V_j}}{e^{U_i^T V_j} + 1} V_j - I_{ij} V_j + \lambda U_i \quad (8)$$

$$\frac{\partial \mathcal{L}}{\partial V_j} = \sum_{i=1}^M \frac{e^{U_i^T V_j}}{e^{U_i^T V_j} + 1} U_i - I_{ij} U_i + \lambda V_j \quad (9)$$

## 4.2 Context-Aware Binomial Retweet Matrix Factorization

As mentioned above, contextual information is an indispensable factor for retweeting prediction due to its effect on users' decisions. Thus, we propose Context-aware Binomial Retweet Matrix Factorization (CBRMF) by considering user community and document clustering learned from contextual information for retweeting behaviors.

**User Community Modeling.** According to sociology and psychology, users under the effect of network structure and information diffusion together gradually form communities corresponding to close social circles or interest groups

with the similar personal preference. In this case, behaviors are more likely to be effected each other in the same social community, e.g., some users like to follow others’s message and are easily influenced by the active neighbors’ decisions.

Based on the above facts, we introduce user community with the hypothesis that the users that are similar in hidden user space have similar personal preferences. We here apply a new community detection algorithm building upon the distance dynamics [21], which automatically spots communities in a network by examining the changes of distances among nodes. Specifically, we first construct an undirected interaction graph  $G = (V, E, W)$ , where  $V$  is the set of users,  $E$  is the set of edges and  $W$  is the corresponding set of weights.  $e = \{u, v\} \in E$  indicates a social relationship between the users  $u$  and  $v$ .  $w(u, v)$  denotes the weight of edge  $e$ . There are various explicit and implicit relations in social networks. For instance, following represents that a user pays close attention to another user, and retweeting reflects that two users appear in the same message with action relevancy. These behaviors can be modeled as the interaction relation graph  $G$ , in which  $w_{ij}$  is the association weight measuring the co-occurrences of users  $u_i$  and  $u_j$ , i.e.  $w_{ij} = \sum(\#follow, \#retweet, \#mention)$  to represent the sum of occurrence times with these actions, where  $\#$  denotes the number of occurrences for users  $u_i$  and  $u_j$  given an action. In particular, we set  $w_{ij} = 1$  in case of the users  $u_i$  and  $u_j$  only connected by following relationship.

After constructing the interaction graph, the next crucial step is to obtain the user communities by performing Attractor method [21]. For the Attractor algorithm, the cohesion parameter  $\lambda$  is used to determine the positive or negative interaction influence on the distances from exclusive neighbors. We use the implementation provided by the authors and the recommended settings as in their paper<sup>1</sup>. Once the clustering is done, we denote user community matrix as  $W \in \mathbb{R}^{M \times M}$ , where  $W_{ij}$  takes the value of 1 if  $C_{u_i}$  and  $C_{u_j}$  belong to the same community and 0 otherwise.

The user communities make different users with the same group become similar in the latent hidden space. Then we can arrive at the following user social community regularizer as

$$\mathcal{L}_1(U) = \|W - g(U^T U)\|_F^2 \quad (10)$$

where the same community for users indicates the two users should be very close and could be large otherwise.

**Document Clustering Modeling.** An empirical observation is the documents with similar content in observed space have similar semantic distance in hidden space, and the similar messages have a high retweeting probability when they have retweeted in the past. However, short text on social networks is very sparse and exists the noisy data, making it hard to find sufficient statistical factors to discover syntactic and semantic dependencies. Here, to alleviate the problem

<sup>1</sup> <https://github.com/YcheCourseProject/CommunityDetection>.

of data sparseness and noisy, we use a collapsed Gibbs sampling method by Dirichlet multinomial mixture (DMM) model for short text clustering, called GSDMM [31]. The model has some good property for the problem of short text clustering, such as fast to converge and cope with the sparse and high dimensional problem of short texts. The model code used are publicly available<sup>2</sup>.

More specifically, we cluster all short texts into different groups by using GSDMM model that documents are similar to one another within the same cluster and are dissimilar to documents in other clusters. After clustering, in our proposed model, we represent document clustering matrix as  $H \in \mathbb{R}^{N \times N}$ , where  $H_{ij}$  takes the value of 1 if  $C_{d_i}$  and  $C_{d_j}$  belong to the same clustering and 0 otherwise.

Similarly, once document clusters are finished, we may arrive at the following document similarity cluster regularizer

$$\mathcal{L}_2(V) = \|H - g(V^T V)\|_F^2 \quad (11)$$

where the same group for documents indicates the latent distance should be very close and could be large otherwise.

**Prediction Approach.** We demonstrate how to construct user community and document clustering regularization terms in the above section. Next, we factorize user and message latent factors with matrices  $W$  and  $H$  collaboratively as

$$\begin{aligned} \mathcal{L}(U, V|R) &= \sum_{i=1}^M \sum_{j=1}^N \ln[g(R_{ij})^{I_{ij}} (1 - g(R_{ij})^{1-I_{ij}})] \\ &\quad - \frac{1}{2\sigma_W^2} \sum_{i=1}^M \sum_{k=1}^M I_{ik}^{(W)} (W_{ik} - g(U_i^T U_k))^2 \\ &\quad - \frac{1}{2\sigma_H^2} \sum_{j=1}^N \sum_{k=1}^N I_{jk}^{(H)} (H_{jk} - g(V_j^T V_k))^2 \\ &\quad - \frac{1}{2\sigma_U^2} \sum_{i=1}^M U_i^T U_i - \frac{1}{2\sigma_V^2} \sum_{j=1}^N V_j^T V_j + Const \end{aligned} \quad (12)$$

where  $I^W$  is a user indicator matrix where  $I_{ij}^W$  is equal to 1 if users  $u_i$  and  $u_j$  belong to the same social group and 0 otherwise, and  $I^H$  is a message indicator matrix where  $I_{ij}^H$  is equal to 1 if message  $m_i$  and  $m_j$  belong to the same topic group and 0 otherwise.

<sup>2</sup> <https://github.com/rwalk/gsdmm>.

Similarly, maximizing the posterior distribution is equivalent to minimizing the sum-of-squared errors function with hybrid quadratic regularization terms:

$$\begin{aligned}
\min_{U,V} \mathcal{L}(U, V|R) &= \sum_{i=1}^M \sum_{j=1}^N [\ln(e^{U_i^T V_j} + 1) - U_i^T V_j I_{ij}] \\
&+ \frac{\alpha}{2} \sum_{i=1}^M \sum_{k=1}^M I_{ik}^{(W)} (W_{ik} - g(U_i^T U_k))^2 \\
&+ \frac{\beta}{2} \sum_{j=1}^N \sum_{k=1}^N I_{jk}^{(H)} (H_{jk} - g(V_j^T V_k))^2 \\
&+ \frac{\lambda}{2} \left( \sum_{i=1}^M \|U_i\|_F^2 + \sum_{j=1}^N \|V_j\|_F^2 \right)
\end{aligned} \tag{13}$$

**Training Model.** Since the objective function is convex with regard to each parameter, a local minimum can be achieved by updating each parameter iteratively. We also directly use SGD method to update the feature vectors  $U_i$  and  $V_j$  given by Eq. (13) as

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial V_j} &= \sum_{i=1}^M \frac{e^{U_i^T V_j}}{e^{U_i^T V_j} + 1} U_i - I_{ij} U_i + \lambda V_j \\
&+ \beta \sum_{k=1}^N I_{jk}^{(H)} g'(V_j^T V_k) (g(V_j^T V_k) - H_{jk}) V_k
\end{aligned} \tag{14}$$

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial V_j} &= \sum_{i=1}^M \frac{e^{U_i^T V_j}}{e^{U_i^T V_j} + 1} U_i - I_{ij} U_i + \lambda V_j \\
&+ \beta \sum_{k=1}^N I_{jk}^{(H)} g'(V_j^T V_k) (g(V_j^T V_k) - H_{jk}) V_k
\end{aligned} \tag{15}$$

where  $g'(x) = \exp(-x)/(1+\exp(-x))^2$  is the derivative of  $g(x)$ . In each iteration,  $U$  and  $V$  are updated based on the latent variables from the previous iteration.

## 5 Experiments

### 5.1 Experimental Settings

We use two real-world social network datasets to evaluate the validity of our proposed model. Weibo is one of the most popular social network platforms in China. In this paper, we use publicly available Weibo dataset [32]. The dataset contains 1,787,443 users, and 300,000 popular messages and 23,755,810 retweetings. In term of messages, we randomly choose 100,000 popular messages and extract the corresponding relationship and retweetings as our experimental dataset. Besides,

we also collect Twitter data using the RESTful APIs with the crawling process from August 10, 2015 to December 10, 2015. The crawl strategy is designed as followings: randomly select 100 users and then collect their tweet lists, the content of each tweet and following relationships among them. Finally, the dataset contains 4,913 users, 275,820 messages and 570,314 retweetings. The basic statistical information is shown in Table 1.

**Table 1.** Statistics of experimental dataset.

Dataset	#Users	#Messages	#Relations	#Retweetings	Sparseness
Weibo	71,649	100,000	1,125,365	7,198,730	0.1%
Twitter	4,913	275,820	1,075,820	570,314	0.04%

We evaluate the quality of the approximate values for retweeting matrix  $R$  using the Root Mean Square Error (RMSE). We can see that a smaller RMSE value means a better performance. Due to the nature of the problem, both the observation and prediction are binary. Hence, we also use the Precision, Recall,  $F_1$  and Accuracy to evaluate the performance of the proposed algorithms. We randomly split each dataset into two disjoint sets, 80% for training and 20% for testing, and perform 5-fold cross validation.

## 5.2 Baseline Methods

We compare our models against the following traditional and state-of-the-art models:

- **PMF.** This method assume that retweetings are from the normal distributions given the factor vectors of users and messages. The user’s retweeting behaviors can be predicted by the inner products of user and message factor vectors based on matrix completion in missing data [20].
- **LRC-BQ.** The method proposes a notion of social influence locality based on pairwise influence and structural diversity, and then uses a logistic regression classifier to predict user’s retweeting behavior [32].
- **MNMF<sub>RP</sub>.** This method utilizes nonnegative matrix factorization to predict retweeting behavior from user and content dimensions, respectively, by using strength of social relationship to constrain objective function [26].
- **SUA-ACNN.** The model proposes a novel attention-based deep neural network to incorporate user’s attention interests and social information for this task by embedding to represent the user, the user’s attention interests, the author and message respectively [34].
- **HCFMF.** This method learns message embedding by jointly taking the message co-occurrence, semantics, social patterns into consideration, then decomposes the user-message matrix and message-message similarity matrix based on a co-factorization model for learning user’s retweeting behavior [25].



We also implement the different configurations of our proposed model to verify the effectiveness of our algorithm.

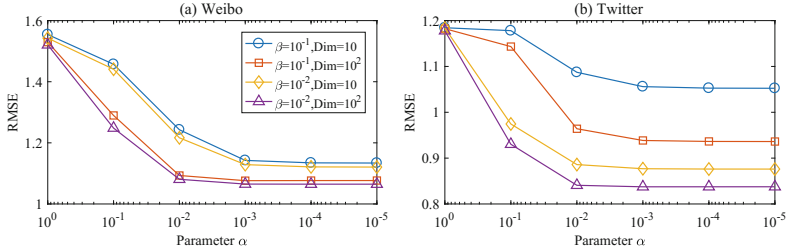


Fig. 3. Impact of  $\alpha$  with CBRMF model on two datasets.

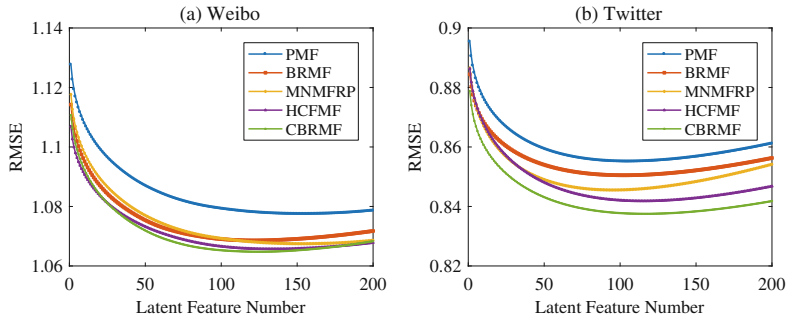


Fig. 4. RMSE vs. latent feature number on two datasets.

- **CBRMF-U**. This method only employs user community factor by eliminating the effect of document clustering regularization term with setting  $\beta = 0$  in Eq. (13).
- **CBRMF-M**. This method only uses document clustering factor by eliminating the effect of user community regularization term with setting  $\alpha = 0$  in Eq. (13).

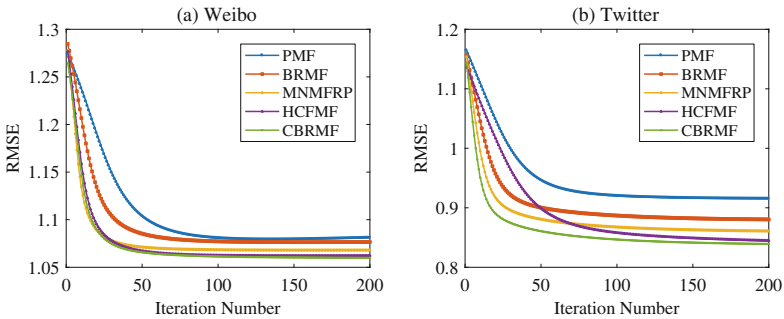
### 5.3 Experimental Results

This section presents some important results settings and also give the results in more details by comparing these baseline methods and discuss the impact of different factors.

**Parameter Settings.** The parameters  $\alpha$  and  $\beta$  provide important contribution strengths for social contextual information in our CBRMF model. The impact of  $\beta$  generally shares the same trend as  $\alpha$ . Hence, we here only illustrate the results of  $\alpha$  due to the space limitation. The experimental results of  $\alpha$  on Weibo and Twitter datasets are shown in Fig. 3. From the figure, we can observe that

the RMSE values gradually decrease while parameter  $\beta$  and dimension of latent features increasing on two datasets, and  $\alpha$  become stable around  $10^{-3}$ . Hence, we set  $\alpha = 10^{-3}$  in our experimental setup. Similarly, we also empirically set the parameters  $\beta = 10^{-3}$  and  $\eta = 10^{-2}$  on two datasets in our models.

**Number of Latent Features.** We perform PMF, BRMF, MNMFRP, HCFMF, and CBRMF models to discover the proper number of latent features on Weibo and Twitter datasets, respectively. The conducted experimental results are shown in Fig. 4 with number of latent features  $K$  from 2 to 200. From these results, we can find that with the latent feature number  $K$  increasing, the RMSE first decreases and then increases, and reach the lowest point around 100. Considering the calculation effect and time efficiency, we choose  $K = 100$  as the dimension of latent feature space.



**Fig. 5.** RMSE vs. Iteration Number on two datasets.

**Number of Iterations.** Similarly, to find the proper number of updating iterations to get a good performance while avoid overfitting, we record the RMSE values for each iteration. Figure 5 illustrates the impacts of the number of iterations on two datasets. From the results, we can see that RMSE values on two datasets decrease gradually with the number of iterations increasing. To reach a converged result with an acceptable computational cost, we set number of iterations to 100 in our proposed models.

**Performance Comparison for Retweeting Prediction.** We demonstrate the prediction performance of our proposed methods and all baseline methods to find who will retweet. Specifically, we run all methods for 5 runs, and report the average results of each method in Table 2. From these results, we can observe the following conclusions: (1) our proposed CBRMF, which incorporates user community and document clustering together, significantly outperforms the baseline methods in our experimental results; (2) the proposed BRMF outperforms PMF, which indicates it is reasonable that retweetings are from binomial distributions instead of normal distributions given the factor vectors of users and messages; (3) the comparison between CBRMF-U vs. CBRMF and CBRMF-M vs. CBRMF, reveals that the user factors and message factors are comparable results

compared with the social factors for retweeting prediction. (4) most of matrix factorization methods such as MNMFRP, HCFMF and CBRMF for retweeting prediction can achieve the accuracy of prediction relatively well. These results suggest that matrix factorization is suitable for the task. (5) SUA-ACNN performs slightly better than most of baseline methods, and also the improvements are statistically significant compared to BRMF. The results demonstrate that the attention-based deep neural network can benefit the performance. We also notice slightly different performance of the two datasets. For example, the BRMF and CBRMF methods achieve better prediction results on Weibo dataset than on Twitter dataset. A possible reason is the average number of retweetings per user on Weibo dataset is much higher than the number on Twitter dataset. In this case, user-based method can generally generate better results since every user has more information to use. This is the possible cause of why the user-based method has better performance. In summary, we conclude that BRMF and CBRMF following binomial distributions are a reasonable assumption, and improve the prediction accuracy by using social contextual information.

**Table 2.** Performance of retweeting prediction with different baseline methods on Weibo and Twitter datasets.

Method	Weibo dataset				Twitter dataset			
	Precision	Recall	$F_1$	Accuracy	Precision	Recall	$F_1$	Accuracy
PMF	0.628	0.607	0.612	0.619	0.611	0.654	0.631	0.621
BRMF	<b>0.669</b>	<b>0.645</b>	<b>0.657</b>	<b>0.668</b>	<b>0.657</b>	<b>0.612</b>	<b>0.635</b>	<b>0.643</b>
LRC-BQ	0.698	0.770	0.733	0.719	0.669	0.638	0.653	0.656
MNMFRP	0.754	0.705	0.729	0.757	0.711	0.688	0.699	0.693
SUA-ACNN	0.746	0.733	0.739	0.753	0.728	0.713	0.720	0.725
HCFMF	0.756	0.742	0.749	0.763	0.739	0.725	0.732	0.735
CBRMF-U	0.723	0.702	0.712	0.723	0.727	0.712	0.719	0.743
CBRMF-M	0.733	0.716	0.724	0.738	0.762	0.751	0.757	0.778
CBRMF	<b>0.802</b>	<b>0.785</b>	<b>0.794</b>	<b>0.797</b>	<b>0.795</b>	<b>0.774</b>	<b>0.784</b>	<b>0.785</b>

**Impact of the Number of Clusters.** The number of clusters with user and message has a great effect on the performance of our proposed CBRMF model. Figure 6 shows the accuracy of retweeting prediction with various values of user and document clusters found on two datasets. From the result, we can see that (1) these datasets have a optimal value for number of clusters when we enlarge the number of cluster; (2) The best number of clusters found by CBRMF is near the latent factor number of groups which means CBRMF can infer the number of clusters automatically when the number of cluster is large enough. Therefore, we can conclude that it is a better practice to set user communities and document clusters to 40 and 80 on Weibo dataset, and 40 and 70 on Twitter dataset, respectively.

**Performance Comparison with Different Community Detection and Short Text Clustering Methods.** We investigate the effects of different

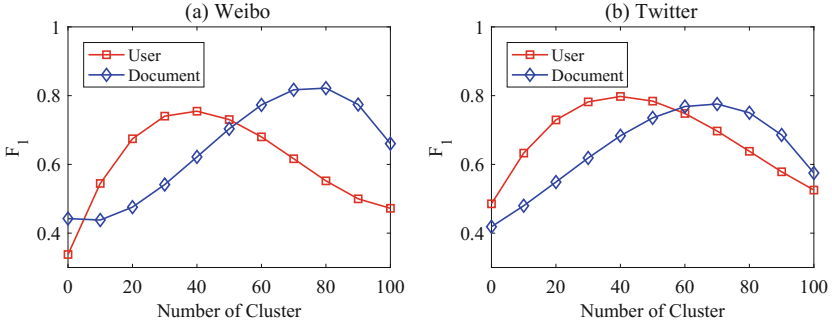


Fig. 6.  $F_1$  vs. different number of cluster on two datasets.

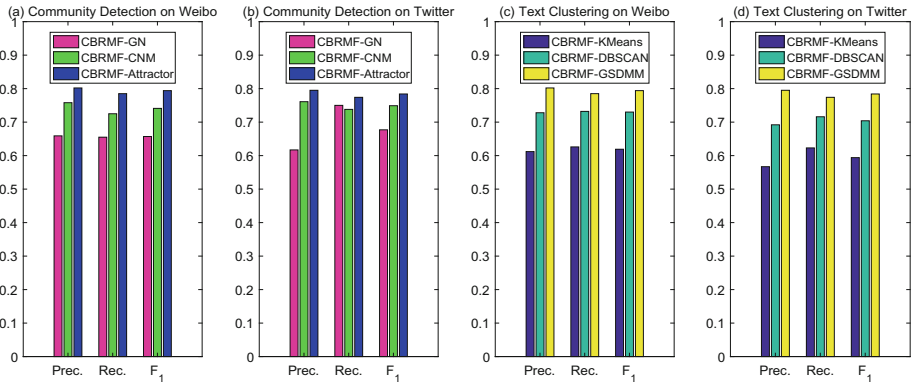


Fig. 7. Performance comparison of CBRMF with different community detection and short text clustering methods on datasets.

methods of community detection and short text clustering in our proposed model. The detailed description is as

- **CBRMF-GN.** The method utilizes the Girvan-Newman algorithm to detect user communities.
- **CBRMF-CNM.** This method uses the Clauset-Newman-Moore community detection method for large networks.
- **CBRMF-KMeans.** This method employs k-means clustering to partition short text documents into clusters.
- **CBRMF-DBSCAN.** This method uses density-based spatial clustering of applications with noise to group short texts documents into clusters.

Figure 7 shows the prediction performance with different community detection and short text clustering methods on Weibo and Twitter datasets. From these results, we can see when choosing Attractor algorithm to perform user’s community detection, CBRMF-Attractor outperforms CBRMF-GN and

CBRMF-CNM on two datasets. We have the similar observation CBRMF-GSDMM performs better than CBRMF-KMeans and CBRMF-DBSCAN over short text clustering while the GSDMM is done. In a word, these results suggest the Attractor and GSDMM models are a better choice for detecting user community and clustering short text documents on social network datasets.

## 6 Conclusion

We propose two novel retweet prediction models based on binomial distributions and contextual information. The proposed two models assume retweetings are from binomial distributions instead of normal distributions under matrix factorization. CBRMF is an extended BRMF model by incorporating user community and document clustering as social regularization terms to alleviate the data sparsity and reduce the noisy information. By experimental evaluation using two real-world social network datasets, we can conclude it is reasonable to assume that retweetings are from binomial distributions, and our proposed methods outperform existing state-of-the-art comparison methods.

**Acknowledgement.** This work is supported by the National Natural Science Foundation of China (No. 61702508, No. 61802404), the National Key Research and Development Program of China (No. 2016QY06X1204), and the Strategic Priority Research Program of the CAS (XDC02000000). This work is also partially supported by Key Laboratory of Network Assessment Technology, Chinese Academy of Sciences and Beijing Key Laboratory of Network security and Protection Technology.

## References

1. Abdullah, N.A., Nishioka, D., Tanaka, Y., Murayama, Y.: User's action and decision making of retweet messages towards reducing misinformation spread during disaster. *J. Inf. Process.* **23**(1), 31–40 (2015)
2. Bae, Y., Ryu, P.-M., Kim, H.: Predicting the lifespan and retweet times of tweets based on multiple feature analysis. *J. Electron. Telecommun. Res. Inst.* **36**(3), 418–428 (2014)
3. Bedi, P., Sharma, C.: Community detection in social networks. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **6**(3), 115–135 (2016)
4. Can, E.F., Oktay, H., Manmatha, R.: Predicting retweet count using visual cues. In: *CIKM*, pp. 1481–1484. ACM (2013)
5. Gao, H., Tang, J., Hu, X., Liu, H.: Content-aware point of interest recommendation on location-based social networks. In: *AAAI*, pp. 1721–1727 (2015)
6. Jamali, M., Ester, M.: A matrix factorization technique with trust propagation for recommendation in social networks. In: *RecSys*, pp. 135–142 (2010)
7. Jiang, B., Sha, Y., Wang, L.: A multi-view retweeting behaviors prediction in social networks. In: Cheng, R., Cui, B., Zhang, Z., Cai, R., Xu, J. (eds.) *APWeb 2015. LNCS*, vol. 9313, pp. 756–767. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-25255-1\\_62](https://doi.org/10.1007/978-3-319-25255-1_62)
8. Jiang, M., et al.: Social contextual recommendation. In: *CIKM*, pp. 45–54 (2012)

9. Lee, K., Mahmud, J., Chen, J., Zhou, M., Nichols, J.: Who will retweet this? Detecting strangers from Twitter to retweet information. *ACM Trans. Intell. Syst. Technol. (TIST)* **6**(3), 31 (2015)
10. Li, C., Bendersky, M., Garg, V., Ravi, S.: Related event discovery. In: *WSDM*, pp. 355–364. ACM (2017)
11. Liu, L., Tang, J., Han, J., Jiang, M., Yang, S.: Mining topic-level influence in heterogeneous networks. In: *CIKM*, pp. 199–208. ACM (2010)
12. Luo, Z., Osborne, M., Tang, J., Wang, T.: Who will retweet me? Finding retweeters in Twitter. In: *SIGIR*, pp. 869–872. ACM (2013)
13. Ma, H., Yang, H., Lyu, M.R., King, I.: SoRec: social recommendation using probabilistic matrix factorization. *Comput. Intell.* **28**(3), 289–328 (2008)
14. Ma, H., King, I., Lyu, M.R.: Learning to recommend with social trust ensemble. In: *SIGIR*, pp. 203–210 (2009)
15. Ma, H., Zhou, D., Liu, C., Lyu, M.R., King, I.: Recommender systems with social regularization. In: *WSDM*, pp. 287–296. ACM (2011)
16. Macedo, A.Q., Marinho, L.B., Santos, R.L.T.: Context-aware event recommendation in event-based social networks. In: *RecSys*, pp. 123–130. ACM (2015)
17. Mahdavi, M., Asadpour, M., Ghavami, S.M.: A comprehensive analysis of tweet content and its impact on popularity. In: *IST*, pp. 559–564. IEEE (2016)
18. Metaxas, P.T., Mustafaraj, E., Wong, K., Zeng, L., O’Keefe, M., Finn, S.: What do retweets indicate? Results from user survey and meta-review of research. In: *ICWSM*, pp. 658–661 (2015)
19. Peng, H.-K., Zhu, J., Piao, D., Yan, R., Zhang, Y.: Retweet modeling using conditional random fields. In: 2011 IEEE 11th International Conference on Data Mining Workshops (ICDMW), pp. 336–343. IEEE (2011)
20. Salakhutdinov, R., Mnih, A.: Probabilistic matrix factorization. In: *NIPS*, pp. 1257–1264 (2007)
21. Shao, J., Han, Z., Yang, Q., Zhou, T.: Community detection based on distance dynamics. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2015, New York, NY, USA*, pp. 1075–1084. ACM (2015)
22. Shi, J., Chen, G., Lai, K.K.: Factors dominating individuals’ retweeting decisions. In: *CyberC*, pp. 161–168. IEEE (2016)
23. Suh, B., Hong, L., Pirolli, P., Chi, Ed.H.: Want to be retweeted? Large scale analytics on factors impacting retweet in Twitter network. In: *SocialCom* (2010)
24. Tang, J., Gao, H., Liu, H.: mTrust: discerning multi-faceted trust in a connected world, pp. 93–102 (2012)
25. Wang, C., Li, Q., Wang, L., Zeng, D.D.: Incorporating message embedding into co-factor matrix factorization for retweeting prediction. In: *IJCNN*, pp. 1265–1272. IEEE (2017)
26. Wang, M., Zuo, W., Wang, Y.: A multidimensional nonnegative matrix factorization model for retweeting behavior prediction. *Math. Probl. Eng.* **2015**, 10 (2015)
27. Wang, X., Lu, W., Ester, M., Wang, C., Chen, C.: Social recommendation with strong and weak ties. In: *CIKM*, pp. 5–14. ACM (2016)
28. Xie, J., Szymanski, B.K.: Towards linear time overlapping community detection in social networks. In: Tan, P.-N., Chawla, S., Ho, C.K., Bailey, J. (eds.) *PAKDD 2012. LNCS (LNAI)*, vol. 7302, pp. 25–36. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-30220-6\\_3](https://doi.org/10.1007/978-3-642-30220-6_3)
29. Yang, C., Liu, L., Jiao, Y., Chen, L., Niu, B.: Research on the factors affecting users’ reposts in microblog. In: *ICSSM*, pp. 1–6. IEEE (2017)

30. Yang, Z., et al.: Understanding retweeting behaviors in social networks. In: CIKM, pp. 1633–1636 (2010)
31. Yin, J., Wang, J.: A Dirichlet multinomial mixture model-based approach for short text clustering. In: KDD, pp. 233–242. ACM (2014)
32. Zhang, J., Tang, J., Li, J., Liu, Y., Xing, C.: Who influenced you? Predicting retweet via social influence locality. *ACM Trans. Knowl. Discov. Data (TKDD)* **9**(3), 25 (2015)
33. Zhang, Q., Gong, Y., Guo, Y., Huang, X.: Retweet behavior prediction using hierarchical Dirichlet process. In: AAAI, pp. 403–409 (2015)
34. Zhang, Q., Gong, Y., Wu, J., Huang, H., Huang, X.: Retweet prediction with attention-based deep neural network. In: CIKM, pp. 75–84. ACM (2016)



# Sparse Gradient Compression for Distributed SGD

Haobo Sun<sup>1</sup>(✉), Yingxia Shao<sup>2</sup>, Jiawei Jiang<sup>4</sup>, Bin Cui<sup>3</sup>, Kai Lei<sup>1</sup>, Yu Xu<sup>4</sup>,  
and Jiang Wang<sup>4</sup>

<sup>1</sup> School of Electronic and Computer Engineering, Peking University,  
Shenzhen, China

{1701213627,leik}@pku.edu.cn

<sup>2</sup> Beijing Key Lab of Intelligent Telecommunications Software and Multimedia,  
BUPT, Beijing, China

shaoyx@bupt.edu.cn

<sup>3</sup> Center for Data Science, Peking University & National Engineering Laboratory  
for Big Data Analysis and Applications, Beijing, China

bin.cui@pku.edu.cn

<sup>4</sup> Tencent Inc., Shenzhen, China

{blue.jwjiang,henrysxu,janwang}@tencent.com

**Abstract.** Communication bandwidth is a bottleneck in distributed machine learning, and limits the system scalability. The transmission of gradients often dominates the communication in distributed SGD. One promising technique is using the gradient compression to reduce the communication cost. Recently, many approaches have been developed for the deep neural networks. However, they still suffer from the high memory cost, slow convergence and serious staleness problems over sparse high-dimensional models. In this work, we propose Sparse Gradient Compression (SGC) to efficiently train both the sparse models and the deep neural networks. SGC uses momentum approximation to reduce the memory cost with negligible accuracy degradation. Then it improves the accuracy with long-term gradient compensation, which maintains global momentum to make up for the information loss caused by the approximation. Finally, to alleviate the staleness problem, SGC updates model weight with the accumulation of delayed gradients at local, called local update technique. The experiments over the sparse high-dimensional models and deep neural networks indicate that SGC can compress 99.99% gradients for every iteration without performance degradation, and saves the communication cost up to 48×.

**Keywords:** Distributed computing system ·  
Distributed optimization algorithm · Gradient compression

## 1 Introduction

Synchronous Stochastic Gradient Descent (SGD) [1–4] has been widely used as a distributed training method for various machine learning models, such as Logistic



Regression [5], Support Vector Machine [6], Graphical Models [7, 8], and Neural Networks [9]. During the training, the workers need to communicate gradients frequently, and this yields a severe bottleneck for scalability, especially when the devices are low-bandwidth [10–12]. Therefore, gradient compression methods are introduced for communication-efficient distributed training.

Gradient sparsification is one effective technique to reduce communication data by only sending large gradients and delaying the transmission of small gradients [13–15]. One example is Gradient Dropping (GD) [14]. It accumulates the gradients in workers, and transmits them when they reach a certain threshold. Another one is Deep Gradient Compression (DGC) [10], which is the state-of-the-art gradient sparsification-based method. It uses momentum correction to deduce the accumulated discounting factor of gradients, thus guaranteeing the accuracy. To alleviate staleness, DGC applies momentum factor masking, which drops the so-called stale gradients.

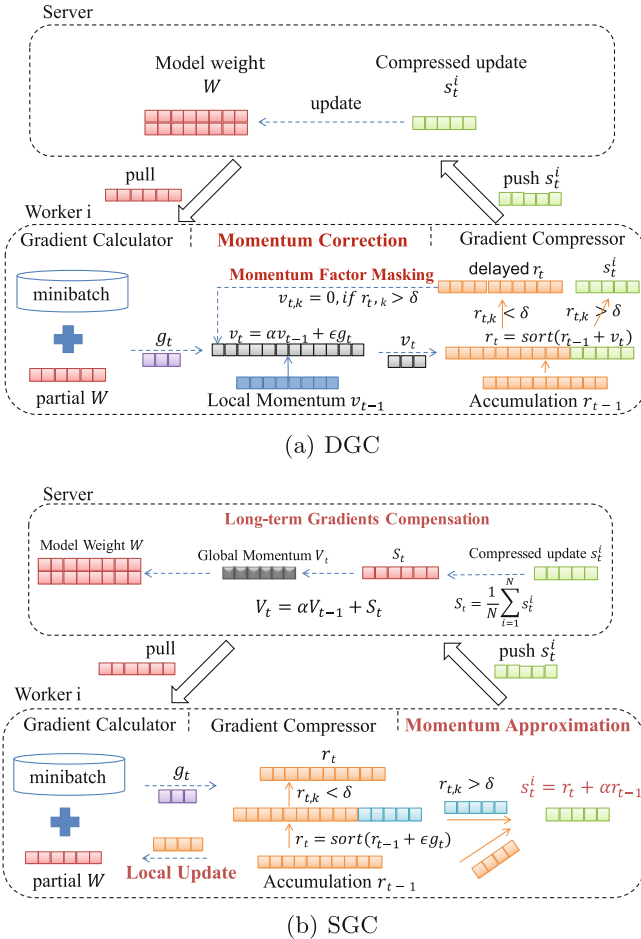


Fig. 1. The overviews of DGC and SGC. Best viewed in color. (Color figure online)

However, the existing gradient sparsification solutions still have the following three problems.

**Expensive Memory Cost on the Sparse High-Dimensional Model.** With large gradient sparsity, most of gradients are delayed, and the size of local accumulation becomes proportion to the number of features. This incurs expensive memory cost when we train sparse high-dimensional models with the existing gradient sparsification solutions. Especially for the DGC, it at least doubles the used memory compared with GD, and makes the memory cost more expensive. As shown in Fig. 1(a), DGC not only stores the accumulation ( $r_t$ ), but also maintains the momentum ( $v_t$ ) in workers. Assume that we train a Logistic Regression model on CTR dataset, which is a private dataset and has around 10 billions different features, DGC will use around 447 GB memory for each worker once the whole feature set is traversed. However, a physical node can hardly afford such huge memory in commodity clusters. In practice, with the limited memory capacity, the expensive memory cost will incur high communication instead.

**Long-Term Gradients Missing.** Reddi [16] has rigorously proven that long-term gradients (i.e., past gradients) can improve the empirical performance and fix the convergence issues. However, in DGC, the momentum factor masking drops the gradients once pushed to the global model, and the long-term values of the dropped gradients are never contributed to accelerating the convergence. GD does not benefit from the long-term gradients as well, because it only accumulates gradients without momentum.

**The Staleness Problem on Sparse Models.** Due to the gradient delay mechanism [13], the local accumulation of gradients on workers is outdated against the global model. The longer gradients delayed, the more inconsistency between the global model and the delayed updates. This is also known as the staleness problem. During the gradient compression, sparse models make the gradients delay for a longer period and aggravate the staleness problem. Although DGC alleviates staleness by momentum factor masking, it fails to maintain the long-term gradients which may help accelerate the convergence.

In this paper, we propose a novel sparse gradient compression method, namely SGC, which addresses all the above problems. Figure 1(b) shows the overview of SGC. It applies *momentum approximation* to ensure the memory efficiency, and also employs *long-term gradient compensation* to improve the convergence performance. Besides, SGC uses *local update* to ease the staleness effect.

We empirically verify our techniques for compressing gradients on both the sparse high-dimensional linear model and the deep neural network. The experimental results demonstrate that SGC is able to compress 99.99% gradients for every iteration without performance degradation. Furthermore, SGC only uses half of the memory compared with DGC, and also reduces the overall communication cost up to  $48\times$ .

In summary, our technique contributions are three-fold:

- We study the relationship between the momentum term and the gradient accumulation under the compression scheme, and propose the momentum approximation technique to reduce memory. Momentum approximation only accumulates gradients at local. When updating the global model, it generates the approximated momentum by ignoring gradients with high-order discounting factors, which only incurs slight accuracy degradation.
- We propose long-term gradient compensation on top of the momentum approximation to improve the accuracy of model performance. Considering that both momentum approximation and momentum factor masking discard the past gradients at local, the long-term gradient compensation uses global momentum to capture the lost gradients, so that the model can still converge fast.
- We apply the local update on each worker to diminish the inconsistency between model and delayed update. Specifically, before calculating the latest gradients with the new model at local, we update the model weight by applying the delayed gradients.
- The experiments clearly showcase that, compared with the standard Momentum SGD, SGC is able to achieve the same convergence when gradient sparsity is 99.99%. As far as we known, SGC is the first compression approach to achieve such a high gradient sparsity.

The remains of the paper are organized as follows: we present the related work and the backgrounds in Sects. 2 and 3 respectively. In Sect. 4, we elaborate our Sparse Gradient Compression with introducing three techniques, which are momentum approximation, long-term gradient compensation, and local update. In the next section, we empirically study the performance of SGC. We conclude our work in Sect. 6.

## 2 Related Work

Reducing the communication data in distributed training is a fundamental problem to achieve high-performance solutions. Many types of research have been conducted on this problem. One basic idea is to compress the gradients to reduce the size of communication data. Gradient quantization and sparsification are two general solutions on top of gradient compression.

*Gradient Quantization.* Seide et al. [17] heuristically quantized each gradient to one bit without accuracy loss. The method emphasized the importance of accumulating the quantization error to guarantee the convergence. Further, Alistarh et al. [18] quantized each gradient to a tuple, which made a trade-off between communication cost and convergence speed. He proved that the compression-variance trade-off is inherent. Wu et al. [19] proposed another quantization-based method, which starts from a similar error feedback scheme, but accumulates both the current gradients and the previous quantization error to narrow the sub-optimal gap. Our proposed SGC is orthogonal to these quantization methods.

*Gradient Sparsification.* Strom et al. [13] proposed to only transmit the gradients that exceed a fixed threshold for compression. They empirically verified that a big fraction of gradients can be safely delayed. Dryden et al. [20] further extended this idea by adopting two approximate adaptive thresholds (positive and negative) and delaying a fixed proportion of gradients. Instead of dropping the positive and negative gradients separately, Aji et al. [14] used a single threshold based on absolute values. They also observed that local thresholds may outperform the global threshold, because parameters have different scales. AdaComp [21] handles the diversity of neural networks, and it tunes the gradient sparsity automatically by grouping the gradients into different bins. Although AdaComp avoids sorting the accumulation to find the threshold, it traverses each bin to find the maximum of the accumulation. Wang et al. [15] dropped gradients by the probability to balance the sparsity and variance. They also appropriately amplified the gradients to ensure the unbiasedness of the sparsity. DGC [10] is the state-of-the-art method. It employs momentum correction and local gradient clipping to improve the local gradient accumulation. It also uses momentum factor masking and warm-up training to overcome the staleness effect. The gradient sparsity of DGC can be 99.9%, which is the highest for deep learning as far as we know. Although most works delayed gradients according to the scale of gradients, Tsuzuku et al. [11] focused on the variance of the gradients and delayed the ambiguous gradients by storing both the gradients and the variances at local.

*Staleness.* Staleness is a common problem leading to performance degradation in the asynchronous training, where workers update the global model weight using outdated gradients. Mitliagkas et al. [22] pointed out that asynchrony leads to staleness, which can be viewed as adding a momentum-like term to the SGD iteration. Wei et al. [23] focused on understanding the effect of stale gradient updates during distributed training, and mitigated the effects of staleness by adjusting the learning rate. Jiang et al. [24] conducted a systematic study that distributed training suffered performance degradation in heterogeneous environments because stragglers harm the convergence, and they used the delayed information of updates to achieve stable convergence.

### 3 Backgrounds

In this section, we introduce the basic concepts of gradient sparsification through reviewing two approaches: Gradient Dropping and Deep Gradient Compression.

#### 3.1 Gradient Dropping

Gradient sparsification reduces the communication bandwidth by delaying the transmission of less important gradients. Gradient Dropping [14], a gradient sparsification-based method, accumulates the delayed gradients locally and transmits them when the accumulated values exceed a threshold. Given the

learning rate  $\epsilon$  and the threshold  $\delta$ , where  $\delta$  is controlled by the input gradient sparsity, Gradient Dropping updates the model weight  $w$  in the  $t^{\text{th}}$  iteration as below:

$$\begin{aligned} g_t &= \nabla f_t(w_{t-1}), \\ r_t &= r_{t-1} + \epsilon g_t, \\ s_t &= \{r_{t,k} \mid r_{t,k} > \delta\}, \\ w_{t,k} &= w_{t-1,k} - s_{t,k}, \end{aligned} \tag{1}$$

where  $g_t$  is the gradient at iteration  $t$ ,  $k$  is the feature index of model weight,  $s_t$  is the set of gradient accumulations that exceed the threshold, and  $r_t$  is the accumulation of delayed gradients. Furthermore,  $r_{t,k}$  which exceeds the threshold is reset to zero after it updates the model weights:

$$r_{t,k} = 0, \text{ if } r_{t,k} > \delta. \tag{2}$$

Assume the gradient on feature index  $k$  has been accumulated for  $T$  iterations before updating the model weights, the change of  $w_{t,k}$  is as follows,

$$w_{t+T,k} = w_{t,k} - \epsilon [g_{t+T,k} + g_{t+T-1,k} + \dots + g_{t+1,k}]. \tag{3}$$

Lin et al. [10] point out that, compared to the Momentum SGD [25], Gradient Dropping ignores the accumulated discounting factor since it simply accumulates the raw gradients, thus leading to the loss of convergence performance.

### 3.2 Deep Gradient Compression

Deep Gradient Compression (DGC) employs four techniques to compress the gradients without loss of model accuracy. Figure 1(a) illustrates the overview of DGC, which is proposed on the basis of Momentum SGD. Momentum SGD is widely used to accelerate training, especially in the presence of high curvature or noisy gradients. It operates as follows:

$$\begin{aligned} \nu_t &= \alpha \nu_{t-1} + \epsilon g_t, \\ w_t &= w_{t-1} - \nu_t, \end{aligned} \tag{4}$$

where  $\nu$  is the momentum that accelerates training and  $\alpha$  is the hyper-parameter of momentum which decays the moving average of past gradients.

In order to reduce the impact on convergence from gradient compression and achieve the same performance as momentum SGD, DGC uses the momentum correction which calculates the compressed momentum on workers, instead of servers. In the  $t^{\text{th}}$  iteration, the momentum correction is executed as below,

$$\begin{aligned} g_t &= \nabla f_t(w_{t-1}), \\ \nu_t &= \alpha \nu_{t-1} + \epsilon g_t, \\ r_t &= r_{t-1} + \nu_t, \\ s_t &= \{r_{t,k} \mid r_{t,k} > \delta\}, \\ w_{t,k} &= w_{t-1,k} - s_{t,k}. \end{aligned} \tag{5}$$

Compared with GD, DGC accumulates the momentum  $v_t$  instead of the raw gradient  $g_t$ . In addition, DGC employs momentum factor masking to alleviate staleness after the delayed momentums update the global model weights,

$$\begin{aligned} r_{t,k} &= 0, \text{ if } r_{t,k} > \delta, \\ v_{t,k} &= 0, \text{ if } r_{t,k} > \delta. \end{aligned} \quad (6)$$

Assume a worker delays the gradient on feature index  $k$  for  $T$  iterations, then the change of  $w_{t,k}$  is as follows,

$$w_{t+T,k} = w_{t,k} - \epsilon \left[ g_{t+T,k} + (1 + \alpha)g_{t+T-1,k} + \cdots + \left( \sum_{\tau=0}^{T-1} \alpha^\tau \right) g_{t+1,k} \right]. \quad (7)$$

Comparing the Eq. 3 and the Eq. 7, DGC deduces the accumulated discounting factor  $\sum_{\tau=0}^{T-1} \alpha^\tau$  for the delayed gradients. Therefore, DGC can alleviate the effect of delaying gradients and achieve a higher gradient sparsity without accuracy loss.

---

**Algorithm 1.** The Pseudocode of SGC

---

**Input:** dataset  $\chi$ , gradient sparsity  $h$

```

1: Initialize:
    $w = \{w[0], w[1], \dots, w[D]\}$ 
    $V = \{V[0], V[1], \dots, V[D]\}$ 
    $r \leftarrow 0; s \leftarrow \emptyset$ 
2: for  $t = 0, 1, \dots$  do
3:   //1. Worker: Gradient Calculation and Compression
4:   Sample a minibatch of examples  $X_t$  from  $\chi$ 
5:    $g_t \leftarrow \text{SGD}(w_{t-1} - r_{t-1}, X_t)$  {Local Update}
6:    $r_t \leftarrow r_{t-1} + \epsilon g_t$ 
7:    $\delta = \text{select\_threshold}(r_t, h)$ 
8:   for  $r_{t,k}$  in  $r_t$  do
9:     if  $r_{t,k} > \delta$  then
10:       $v_{t,k} \leftarrow r_{t,k} + \alpha r_{t-1,k}$  {Momentum Approximation}
11:       $r_{t,k} = 0$ 
12:       $s_t \leftarrow s_t \cup \{v_{t,k}\}$ 
13:     end if
14:   end for
15:    $\text{transmit}(s_t)$ 
16:
17:   //2. Server: Model Update
18:    $\text{receive}(s_t)$ 
19:    $S_t \leftarrow \text{average}(s_t)$ 
20:    $V_t \leftarrow \alpha V_{t-1} + S_t$  {Long-term Gradient Compensation}
21:    $w_t \leftarrow w_{t-1} - V_t$ 
22: end for

```

---

**Memory Cost Analysis.** However, from Eq. 5, we find that DGC needs to maintain both the momentum  $\nu$  and the accumulation  $r$  on a single worker. The memory used in the workers is at least twice larger than that of Gradient Dropping. In other words, DGC sacrifices the memory to achieve high model accuracy. As briefly discussed in the Introduction, when using DGC to train a sparse high-dimensional model, a commodity cluster cannot afford sufficient memory for each worker. In this paper, we focus on proposing a novel gradient compression method, which entails the same memory cost as GD, while achieving similar (or even better) model accuracy compared with DGC.

## 4 Sparse Gradient Compression

Figure 1(b) shows the logical overview of Sparse Gradient Compression (SGC). In each worker, SGC employs the *momentum approximation* technique, which accumulates the delayed gradients with small memory and generates approximated momentum. Then, in the servers, SGC uses the *long-term gradient compensation* to maintain past gradients and obtain high model accuracy. Moreover, to alleviate the staleness effect, SGC uses the *local update* in each worker to diminish the inconsistency between the global model and delayed update. Algorithm 1 illustrates the pseudo code of SGC.

### 4.1 Momentum Approximation

As Mitliagkas et al. [22] claim, the model of asynchrony results in a form of implicit momentum compared with the explicit momentum  $\nu$ . The claim implies that the acceleration of explicit momentum with small absolute magnitude can be ignored. Actually, DGC also drops the momentums with stale information on the workers via momentum factor masking. Following a similar idea, we propose to achieve the momentum approximation by ignoring the high-order discounting factors which are decayed too much. Instead of accurately calculating the momentum  $\nu_t$  as shown in Eq. 5, SGC uses the first-order momentum for approximation, and in each iteration, it computes as below,

$$\begin{aligned} g_t &= \nabla f_t(w_{t-1}), \\ r_t &= r_{t-1} + \epsilon g_t, \\ s_t &= \{r_{t,k} + \alpha r_{t-1,k} \mid r_{t,k} > \delta\}, \\ w_{t,k} &= w_{t-1,k} - s_{t,k}. \end{aligned} \tag{8}$$

Note that the term  $\alpha r_{t-1}$  is the approximation of momentum  $\nu_t$ . Compared with GD, SGC transmits the first-order momentum instead of the gradients.

We also reset  $r_{t,k}$  to zero at local after it is applied to the model update,

$$r_{t,k} = 0, \text{ if } r_{t,k} > \delta. \tag{9}$$

After  $T$  iterations of local accumulation, the change of  $w_{t,k}$  is:

$$w_{t+T,k} = w_{t,k} - \epsilon [g_{t+T,k} + (1 + \alpha)g_{t+T-1,k} + \cdots + (1 + \alpha)g_{t+1,k}]. \tag{10}$$

From the Eqs. 7 and 10, it is easy to figure out that the momentum approximation actually discards the high-order discounting factor of past gradients. With such approximation, SGC only needs to store the  $r_t$  on each worker, and entails the same memory cost with GD, but it speeds up the convergence with momentum. Compared with DGC, the empirical study will demonstrate that in practice the momentum approximation obtains the similar model performance.

## 4.2 Long-Term Gradient Compensation

Both momentum approximation and DGC employ momentum factor masking to relieve the staleness. However, this also overlooks the acceleration of past gradients which may not be stale. Reddi et al. [16] have pointed out that long-term gradients can improve empirical performance and fix the convergence issues. Therefore, to compensate for these missing information and achieve high model accuracy, SGC applies *long-term gradients compensation* which maintains long-term gradients globally (in servers). To model how the past gradients affect the updating direction of the current global model, we simply present the long-term gradients in the form of global momentum.

Without gradient compression, we can maintain the ideal global momentum  $V_t$  by averaging the gradients from the  $N$  workers as follows,

$$\begin{aligned} G_t &= \frac{1}{N} \sum_{i=1}^N g_t^i, \\ V_t &= \alpha V_{t-1} + \epsilon G_t, \\ w_t &= w_{t-1} - V_t. \end{aligned} \tag{11}$$

Now considering the gradient compression in SGC, we revise the ideal global momentum for the momentum approximation as follows,

$$\begin{aligned} S_t &= \frac{1}{N} \sum_{i=1}^N s_t^i, \\ V_t &= \alpha V_{t-1} + S_t, \\ w_t &= w_{t-1} - V_t. \end{aligned} \tag{12}$$

**Theoretical Analysis.** Long-term gradient compensation makes up for the information loss caused by the momentum approximation and momentum factor masking in SGC. Here we give a theoretical analysis in a simplified version of gradient sparsification. Assume that the delayed momentum is transmitted for every  $T$  iterations, and the transmission has been taken  $K$  times, we proceed to compare the momentum  $v_{KT+1}$  in the  $(KT+1)$ -th iteration of SGC, DGC and Momentum SGD [25] as follows. For the Momentum SGD, the momentum  $v_{KT+1}$  is

$$v_{KT+1} = \sum_{k=0}^{K-1} \sum_{t=1}^T \alpha^{T-t+K-k} g_{kT+t} + g_{KT+1}. \tag{13}$$

And the momentum  $v_{KT+1}$  in DGC with momentum factor masking is

$$v_{KT+1} = g_{KT+1}. \tag{14}$$



Compared with the Eq. 13, DGC loses the past gradients from  $g_1$  to  $g_{KT}$ . In other words, Momentum SGD with momentum factor masking degenerates to vanilla SGD under the worst case scenario.

With the long-term gradient compensation, the momentum in SGC is as follows:

$$v_{KT+1} = \sum_{k=0}^{K-1} \sum_{t=1}^T (1 + \alpha) \alpha^{K-k} g_{kT+t} + g_{KT+1}, \quad (15)$$

which preserves the past gradients as Momentum SGD does. In addition, comparing the Eq. 13 and the Eq. 15, the global momentum places more emphasis on the past gradients (i.e.,  $(1 + \alpha) \alpha^{K-k} > \alpha^{T-t+K-k}$ ), which suggests that we should decrease the hyper-parameter of the momentum for SGC. However, according to our experiments, we observe that SGC can obtain a similar performance as the Momentum SGD even if choosing the same hyper-parameter.

### 4.3 Local Update for the Staleness Effect

Parameter staleness is a common phenomenon in distributed training, which might update the model in the wrong direction [23, 26]. The gradient compression delays the update of the model, and incurs the staleness problem, which leads to performance degradation. When training the sparse high-dimensional model, the staleness effect becomes even worse because the gradients may be delayed for a longer period.

The key observation for compression scheme is that the latest model weights pulled from servers lack the information of delayed gradients, which are locally compressed as an accumulated value. Nesterov et al. [27] calculate the gradient after updating the model weight with momentum, aiming at avoiding the momentum being too large to jump over the optimality. Inspired by this work, we update model weights with  $r$  locally, and then compute the latest gradients  $g_t$  according to the updated model. We formalize the computation as below:

$$\begin{aligned} w_{t-1} &= w_{t-1} - r_{t-1}, \\ g_t &= \nabla f(w_{t-1}). \end{aligned} \quad (16)$$

By considering the information of delayed momentum  $r_t$ , the latest gradients  $g_t$  do revise the wrong directions may be introduced by the stale accumulation at local. Actually, the *local update* is orthogonal to the previous techniques [10] for staleness. The prior works all ignore making use of the advance update to relieve the staleness effect caused by gradient compression.

## 5 Experiments

In this section, we validate our approach for both convex optimization and non-convex optimization respectively, i.e., analyze the effectiveness of our proposed SGC algorithm on  $\ell_2$ -regularized Logistic Regression and the deep neural network. Then we evaluate the advantages of the techniques used by SGC separately. At last, we demonstrate the superiority of SGC for the sparse high-dimensional model by presenting the memory cost and communication time.

## 5.1 Experimental Settings

*Benchmarks.* We choose  $\ell_2$ -regularized Logistic Regression (LR) as the sparse model, and Factorization-machine supported Neural Networks (FNN) [28] as Deep Neural Network respectively.

*Environments.* Both models are implemented on the parameter server (PS) [29, 30], a common distributed ML framework. PS consists of two roles—workers and servers, where workers calculate the gradients of model given a subset of the training data, and servers store the global ML model. And there are three steps on workers inside one iteration of SGD—pull the current model (weight vector) from servers, compute the gradients with a minibatch, and push the gradients to the servers. The PS runs on a cluster, where each physical machine is equipped with 128 GB RAM, 48 cores and 10 GB Ethernet. In addition, for LR, every worker uses 12 GB RAM and 4 cores, and each server uses 100 GB RAM and 40 cores. For FNN, each worker uses 18 GB RAM and 6 cores, and each server uses 100 GB RAM and 40 cores.

**Table 1.** Dataset statistics

Dataset	Size	Dimension	#non-zero-features
KDD2010	2.5 GB	20.2M	19.3M
URP	169.7 GB	400M	100M
CTR	3.1 TB	100G	10G

*Datasets.* Table 1 summarizes the statistics of datasets used in the experiments. KDD2010 is the public dataset for predicting student performance on mathematical problems. CTR is a large-scale proprietary click-through rate dataset, and URP is a user-response-prediction dataset. The CTR and URP are two internal datasets from our industrial partner. We train LR model on KDD2010 and CTR, and train FNN model on URP. The FNN consists of one embedding layer and four full-connected layers, and we apply the gradient compression on the full-connected layers which have more parameters than the embedding layer.

*Compared Methods and Parameter Settings.* SGC is the proposed method with *momentum approximation*, *long-term gradient compensation*, and *local update*. SGC-MA is the method only applying momentum approximation, and SGC-MA-LG is the method applying momentum approximation and long-term gradient compensation. Further, we compare SGC with three competitors. One is the momentum SGD without compression, named Baseline. The other two are Gradient Dropping (GD) [14] and DGC [10].

All the gradient compression methods use the same hyper-parameters as the Baseline, except the gradient sparsity. **Gradient sparsity** is defined as the

**Table 2.** The AUC and LOGLOSS of various methods on KDD2010, CTR and URP datasets.

		Baseline	GD	GD-LU	DGC	DGC-LU	SGC-MA	SGC-MA-LG	SGC
KDD2010 (50 epochs)	Auc	85.30%	84.57%	84.83%	84.74%	85.20%	84.72%	85.37%	<b>85.58%</b>
	Logloss	0.2996	0.3040	0.3025	0.3032	0.3004	0.3033	0.2978	<b>0.2966</b>
CTR (50 epochs)	Auc	73.30%	72.63%	72.92%	73.19%	73.27%	73.16%	73.33%	<b>73.40%</b>
	Logloss	0.4063	0.4097	0.4076	0.4066	0.4054	0.4069	0.4053	<b>0.4047</b>
URP (30 epochs)	Auc	72.78%	72.59%	72.68%	72.71%	72.82%	72.68%	72.95%	<b>72.98%</b>
	Logloss	0.3577	0.3585	0.3581	0.3578	0.3573	0.3580	0.3567	<b>0.3566</b>

number of delayed parameters divided by the total number of parameters in one iteration. To show the compression efficiency, we try to choose the gradient sparsity as high as possible until the convergence speed becomes degenerated compared with the Baseline. We will present the gradient sparsity in the corresponding experiments.

*Metrics.* We evaluate the effectiveness of gradient compression algorithms with two metrics: Area Under the Curve (AUC) and logistic loss (LOGLOSS). AUC is equal to the probability that a classifier ranks a randomly chosen positive instance higher than a randomly chosen negative one. LOGLOSS measures the performance of a classification model where the prediction is a probability value between 0 and 1.

## 5.2 Effectiveness Comparison

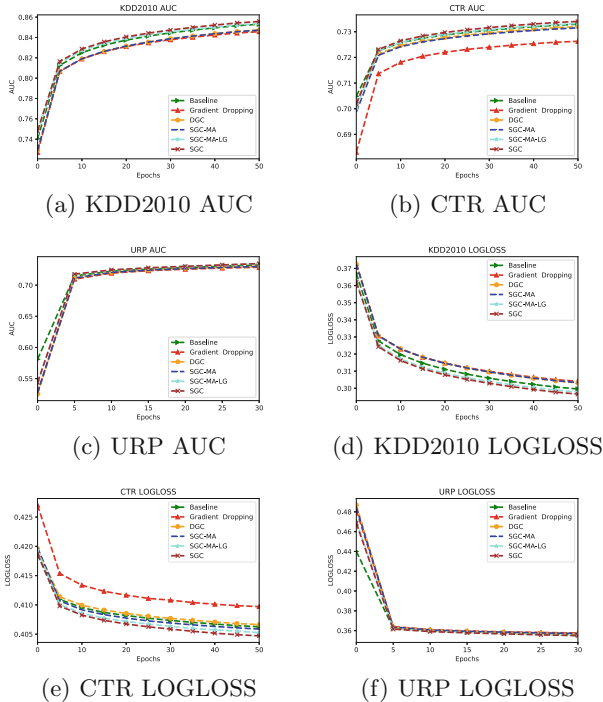
In Fig. 2, we visualize the AUC and LOGLOSS of comparing SGC, SGC-MA, SGC-MA-LG, DGC and GD on all the three datasets. Table 2 summarizes the AUC and LOGLOSS of the final model. For CTR and URP, we set the gradient sparsity to 99.99%. For KDD2010, the gradient sparsity is 99.88%, because the size of gradients for a worker in one iteration is really small (e.g., only several thousand) and the higher gradient sparsity leads to the model being divergent. Besides, LR runs 50 epochs, and FNN uses 30 epochs.

*Comparison of SGC, Baseline, DGC, and GD.* First, we discuss the performance of LR on KDD2010 and CTR as the sparse case. Figures 2(a) and (d) show the AUC and LOGLOSS of KDD2010, while Figs. 2(b) and (e) present the results of CTR. It is clear that SGC has the best convergence performance (both AUC and LOGLOSS) across different datasets during the whole training. For instance, on KDD2010, it takes 27 epochs for SGC to reach 84.50% AUC, while Baseline, GD and DGC need 31, 47, 41 epochs, respectively. On the other hand, both GD and DGC converge more slowly than the Baseline because of the staleness effect and the missing long-term gradients. In addition, SGC solves the problem with

local update and long-term gradient compensation, and it can achieve better performance than the Baseline. Table 2 shows, on CTR, SGC reaches 73.40% AUC, while Baseline has the AUC of 72.63%.

Second, we study the effectiveness of FNN on URP with various gradient compression methods. Different from the sparse case, the whole model weights are pulled by each worker, therefore the gradient size of the full-connected layer is fixed and each gradient is traversed. Figures 2(c) and (f) are the AUC and LOGLOSS of URP. There is no significant difference between GD and DGC, both of which show slight convergence degradation compared with the Baseline because most of the gradients are delayed. In the early stage of training, all the compression methods converge slower than the Baseline, but SGC catches up soon for 3 epochs and achieves better performance on AUC (+0.20%) when executing 30 epochs.

In summary, SGC offers significant improvement on the sparse high-dimension dataset. It also achieves higher gradient sparsity on Deep Neural Network.



**Fig. 2.** Training results of various methods on KDD2010, CTR and URP datasets.

*Comparison of SGC-MA, SGC-MA-LG and SGC.* Here we analyze the effects of the three techniques (i.e., *momentum approximation*, *long-term gradient compensation*, and *local update*) in SGC respectively. Figures 2(a), (b), (d) and (e)

illustrate the results of the sparse high-dimensional model. First, we observe that DGC and SGC-MA are almost indistinguishable on the convergence speed. For example, the difference of LOGLOSS between them is around 0.02%. This implies that the drop of high-order discounting factors does not degrade the performance significantly. Second, with the long-term gradients, SGC-MA-LG is much faster and reaches better performance than SGC-MA. Finally, with the technique of local update, SGC performs the best against SGC-MA and SGC-MA-LG. Furthermore, we also demonstrate that *local update* is an orthogonal optimization technique, and can improve the performance of other gradient compression solutions. We ran GD-LU and DGC-LU on the three datasets. Table 2 shows that local update decreases the AUC degradation of GD from 0.73% to 0.47% on KDD2010, and decreases the performance loss of DGC from 0.11% to 0.03% on CTR.

In Figs. 2(c) and (f), all the solutions have similar performance on URP with FNN. The results demonstrate that SGC and its corresponding optimization techniques also guarantee the convergence performance on Deep Neural Networks.

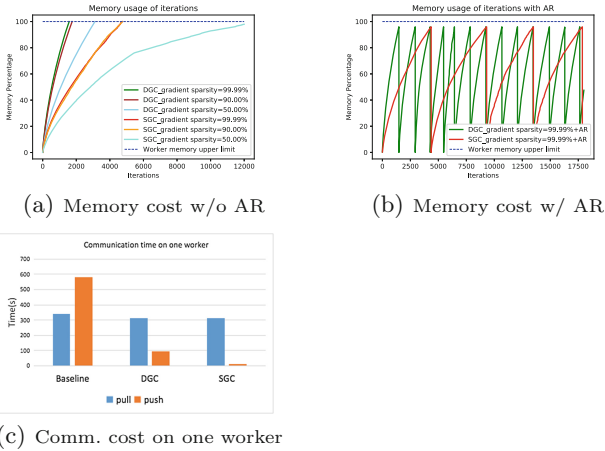


Fig. 3. Results of efficiency comparison

### 5.3 Efficiency Comparison

*Memory Cost.* We compare the memory usage of DGC and SGC on CTR to illustrate the advantage of SGC for the sparse high-dimensional model. Before presenting the results, we introduce an Accumulation Removing (AR) operation to make the compression techniques more reliable in practice. AR is an operation which mandatory pushes the accumulation to the servers when the memory usage of the accumulation is close to the maximum capacity of a worker. Although this may harm the benefit of gradient compression because of occupying the

network bandwidth, it guarantees the systematic reliability of all the gradient compression techniques for the extremely large data set, like CTR.

Figure 3(a) shows the memory usage of a worker without Accumulation Removing (AR). It is easy to figure out that DGC is  $3\times$  faster than SGC to reach the memory budget. For instance, when the gradient sparsity is 99.99%, the accumulation size of DGC reaches the upper limit quickly for about 1600 iterations, while SGC requires around 4700 iterations to reach the same limit. Because with the help of momentum approximation, SGC uses at most half-memory of the one DGC does. Figure 3(b) shows the memory usage with AR. With the help of AR, all algorithms can be executed successfully, and the AR operation improves the reliability of gradient compression solutions. And we also observe that DGC executes AR more frequently ( $3\times$ ) than SGC, and this leads to increasing the communication time.

*Communication Cost.* Finally, we compare the communication time among different gradient compression methods. Figure 3(c) is the communication cost of a single worker for one epoch. Compared with the Baseline, both of the gradient compression techniques reduce the communication cost of push operation. The push operation of Baseline takes about 582s, while DGC and SGC only takes 95s (i.e.,  $6\times$  faster) and 12s (i.e.,  $48\times$  faster), respectively. However, DGC is about  $8\times$  slower than SGC, despite using the same gradient sparsity. The reason is that DGC needs more AR to reduce the memory overhead of workers, while SGC alleviates the large memory footprints via momentum approximation.

## 6 Conclusion

In this paper, we introduced a new gradient compression approach, called Sparse Gradient Compression (SGC), for efficiently training the sparse models with distributed SGD. To reduce the memory cost of accumulate gradients, we proposed momentum approximation. Then by designing long-term gradient compensation, SGC can effectively make up for the missing of acceleration information brought by the approximation and momentum factor masking. Furthermore, we applied local update to ease the staleness effect. We also theoretically analyzed the momentum convergence behavior under the new compression scheme. Experimental results on the sparse high-dimensional model and deep neural network demonstrated the efficiency of our proposed solution.

**Acknowledgements.** This work is supported by the National Key Research and Development Program of China (No. 2018YFB1004403), NSFC (No. 61832001, 61702015, 61702016, 61572039), and PKU-Tencent joint research Lab.

## References

1. Bottou, L.: Large-scale machine learning with stochastic gradient descent. In: COMPSTAT'2010, pp. 177–186. Physica-Verlag HD (2010)
2. Li, Y., Chen, Z., Cai, Y., Huang, D., Li, Q.: Accelerating convolutional neural networks using fine-tuned backpropagation progress. In: Bao, Z., Trajcevski, G., Chang, L., Hua, W. (eds.) DASFAA 2017. LNCS, vol. 10179, pp. 256–266. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-55705-2\\_20](https://doi.org/10.1007/978-3-319-55705-2_20)
3. Zhao, K., Zhang, J., Zhang, L., Li, C., Chen, H.: CDSFM: a circular distributed SGLD-based factorization machines. In: Pei, J., Manolopoulos, Y., Sadiq, S., Li, J. (eds.) DASFAA 2018. LNCS, vol. 10828, pp. 701–709. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-91458-9\\_43](https://doi.org/10.1007/978-3-319-91458-9_43)
4. Wang, K., Peng, H., Jin, Y., Sha, C., Wang, X.: Local weighted matrix factorization for top-n recommendation with implicit feedback. *Data Sci. Eng.* **1**(4), 252–264 (2016)
5. Davis, L.J., Offord, K.P.: Logistic regression. *J. Pers. Assess.* **68**(3), 497–507 (1997)
6. Hearst, M.A., Dumais, S.T., Osuna, E., Platt, J., Scholkopf, B.: Support vector machines. *IEEE Intell. Syst. Appl.* **13**(4), 18–28 (1998)
7. Jiang, J., Zhang, Z., Cui, B., Tong, Y., Xu, N.: StroMAX: partitioning-based scheduler for real-time stream processing system. In: Candan, S., Chen, L., Pedersen, T.B., Chang, L., Hua, W. (eds.) DASFAA 2017. LNCS, vol. 10178, pp. 269–288. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-55699-4\\_17](https://doi.org/10.1007/978-3-319-55699-4_17)
8. Bhuiyan, M., Hasan, M.A.: Representing graphs as bag of vertices and partitions for graph classification. *Data Sci. Eng.* **3**(2), 150–165 (2018)
9. Schmidhuber, J.: Deep learning in neural networks: an overview. *Neural Netw.* **61**, 85–117 (2015)
10. Lin, Y., Han, S., Mao, H., Wang, Y., Dally, B.: Deep gradient compression: reducing the communication bandwidth for distributed training. In: ICLR (2018)
11. Tsuzuku, Y., Imachi, H., Akiba, H.: Variance-based gradient compression for efficient distributed deep learning. arXiv preprint [arXiv:1802.06058](https://arxiv.org/abs/1802.06058) (2018)
12. Jiang, J., Fu, F., Yang, T., Cui, B.: SketchML: accelerating distributed machine learning with data sketches. In: SIGMOD, pp. 1269–1284. ACM (2018)
13. Strom, N.: Scalable distributed DNN training using commodity GPU cloud computing. In: INTERSPEECH (2015)
14. Aji, A.F., Heafield, K.: Sparse communication for distributed gradient descent. In: EMNLP, pp. 440–445. Association for Computational Linguistics (2017)
15. Wangni, J., Wang, J., Liu, J., Zhang, T.: Gradient sparsification for communication-efficient distributed optimization. arXiv preprint [arXiv:1710.09854](https://arxiv.org/abs/1710.09854) (2017)
16. Reddi, S.J., Kale, S., Kumar, S.: On the convergence of adam and beyond. In: ICLR (2018)
17. Seide, F., Fu, H., Droppo, J., Li, G., Yu, D.: 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. In: INTERSPEECH (2014)
18. Alistarh, D., Grubic, D., Li, J., Tomioka, R., Vojnovic, M.: QSGD: communication-efficient SGD via gradient quantization and encoding. In: NIPS, pp. 1709–1720. Curran Associates Inc. (2017)
19. Wu, J., Huang, W., Huang, J., Zhang, T.: Error compensated quantized SGD and its applications to large-scale distributed optimization. In: ICML, pp. 5321–5329 (2018)

20. Dryden, N., Moon, T., Jacobs, S.A., Essen, B.V.: Communication quantization for data-parallel training of deep neural networks. In: MLHPC, pp. 1–8. IEEE (2016)
21. Chen, C.-Y., Choi, J., Brand, D., Agrawal, A., Zhang, W., Gopalakrishnan, K.: Adacomp: adaptive residual gradient compression for data-parallel distributed training. In: AAAI, pp. 2827–2835 (2018)
22. Mitliagkas, I., Zhang, C., Hadjis, S., Ré, C.: Asynchrony begets momentum, with an application to deep learning. In: Allerton, pp. 997–1004. IEEE (2016)
23. Zhang, W., Gupta, S., Lian, X., Liu, J.: Staleness-aware async-SGD for distributed deep learning. In: IJCAI, pp. 2350–2356. AAAI Press (2016)
24. Jiang, J., Cui, B., Zhang, C., Yu, L.: Heterogeneity-aware distributed parameter servers. In: SIGMOD, pp. 463–478. ACM (2017)
25. Polyak, B.T.: Some methods of speeding up the convergence of iteration methods. *USSR Comput. Math. Math. Phys.* **4**(5), 1–17 (1964)
26. McMahan, B., Streeter, M.: Delay-tolerant algorithms for asynchronous distributed online learning. In: NIPS, pp. 2915–2923 (2014)
27. Nesterov, Y., et al.: Gradient methods for minimizing composite objective function (2007)
28. Zhang, W., Du, T., Wang, J.: Deep learning over multi-field categorical data. In: Ferro, N., et al. (eds.) ECIR 2016. LNCS, vol. 9626, pp. 45–57. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-30671-1\\_4](https://doi.org/10.1007/978-3-319-30671-1_4)
29. Jiang, J., Lele, Y., Jiang, J., Liu, Y., Cui, B.: Angel: a new large-scale machine learning system. *Natl. Sci. Rev.* **5**(2), 216–236 (2017)
30. Lele, Y., Zhang, C., Shao, Y., Cui, B.: LDA\*: a robust and large-scale topic modeling system. *Proc. VLDB Endowment* **10**(11), 1406–1417 (2017)





# STDR: A Deep Learning Method for Travel Time Estimation

Jie Xu<sup>(✉)</sup>, Yong Zhang, Li Chao, and Chunxiao Xing

Department of Computer Science and Technology,  
Research Institute of Information Technology,  
Beijing National Research Center for Information Science and Technology,  
Institute of Internet Industry, Tsinghua University, Beijing 100084, China  
xuj15@mails.tsinghua.edu.cn, {zhangyong05, li-chao,  
xingcx}@tsinghua.edu.cn

**Abstract.** With the booming traffic developments, estimating the travel time for a trip on road network has become an important issue, which can be used for driving navigation, traffic monitoring, route planning, and ride sharing, etc. However, it is a challenging problem mainly due to the complicate spatial-temporal dependencies, external weather conditions, road types and so on. Most traditional approaches mainly fall into the sub-segments or sub-paths categories, in other words, divide a path into a sequence of segments or sub-paths and then sum up the sub-time, yet which don't fit the real-world situations such as the continuously dynamical changing route or the waiting time at the intersections. To address these issues, in this paper, we propose an end to end Spatial Temporal Deep learning network with Road type named STDR to estimate the travel time based on historical trajectories and external factors. The model jointly leverages CNN and LSTM to capture the complex nonlinear spatial-temporal characteristics, more specifically, the convolutional layer extracts the spatial characteristics and the LSTM with attention mechanism extracts the time series characteristics. In addition, to better discover the influence of the road type, we introduce a road segmentation approach which is capable of dividing the trajectory based on the shape of trajectory. We conduct extensive verification experiments for different settings, and the results demonstrate the superiority of our method.

**Keywords:** Travel time estimation · CNN · LSTM · Road network

## 1 Introduction

Nowadays, with the explosive growth of the location-enabled devices, the importance and usage of geospatial information have attracted more and more attention from researchers in many applications. In this paper, we mainly focus on the estimation of travel time, which can bring societal and environmental benefits, and is useful for driving navigation, traffic monitoring, route planning, ride sharing and so on [1–5]. However, evaluating an accurate travel time is a challenging problem affected by the following aspects: (1) The road traffic condition continuously dynamically changes during the process of vehicles moving. (2) The driver needs to slow down or wait for a

while at the intersection, and the waiting time is a random variable, modeling the waiting time is not easy. (3) Different road segments may exhibit very different behaviors due to external requirements, for instance, residential areas speed limits are distinct from industrial areas. Although the problem has been widely studied in the past, however, traditional travel time estimation approaches mostly adopt divide-and-conquer approach [6–8]. Those methods mainly decompose a path into a sequence of sub-segments or sub-paths, and then sum up the multiple sub-segments or sub-paths travel time into a whole. Nonetheless, those methods are model-driven and can't handle the delay time caused by the turnings or intersections very well, and the estimated travel time errors, which are affected by external factors such as weather or road types, will also accumulate.

Generally speaking, solving this kind of optimal problem is not easy. Fortunately, recently deep learning has achieved considerable achievements in computer vision, machine translation, image generation, natural language processing field and road trajectories [9, 10]. Deep learning approaches have a strong capability to learn more latent features and simulate complicated dynamics trajectory problem [4, 11–13]. Motivated by aforementioned knowledge, in this paper, we propose an end-to-end Spatial Temporal Deep learning network with Road types (STDR) by using convolutional neural networks (CNN) and long short-term memory (LSTM) to jointly capture complex spatial and temporal nonlinear correlations. The core idea lies in transforming the trajectory data into vector space, and applying the neural network on them. The contributions of this work are summarized as follows.

First, to capture the spatial features of a trajectory, we embed the GPS points into corresponding vectors rather than working on them directly. The vectors can preserve the original correlation of different points, and then are fed into the CNN to learn the spatial dependencies.

Second, we introduce LSTM with attention mechanism to model the time series. The attention mechanism can judge which segment has a higher weight for estimating the whole trajectory. Meanwhile, the influences of external factors (e.g., weather and time metadata) are also concentrated into the LSTM input, which can significantly improve the predicting accuracies.

Third, since the trajectory sampling frequency is fixed, the travel distances with different velocities at the same time interval are diverse. For example, the driving distance on the highway is longer than the distance on an overpass at the same time interval. Therefore, we partition the trajectory into many sub-segments based on the road types, then the sub-segments are encoded into vectors and concatenated with the output of LSTM for training the model together.

Fourth, extensive comprehensive experiments on the real datasets are conducted, and results show the advancement of our approach.

The remainder of this paper is structured as follows. Section 2 describes a review of literature, and Sect. 3 introduces the details of our algorithm. We conduct experiments and evaluate the results in Sect. 4. Finally, Sect. 5 concludes the paper.

## 2 Related Work

Existing researches can be roughly classified into two categories, i.e., the traditional approaches with sub-segments or sub-paths, and the deep learning approaches. In this section, we introduce the literature and summarize the key technologies.

### 2.1 Traditional Travel Time Prediction Approaches

Wang et al. [7] proposed an efficient and simple model that leverages plenty of historical trips without using the intermediate trajectory points to evaluate the travel time between source and destination. Comparing with the most existing approaches, it retrieves the neighboring trips with a similar origin and destination to approximately estimate the travel time. However, the method outperformed many online methods. To deal with traffic time series which were usually sparse, dependent and heterogeneous (e.g., some segments may have morning and afternoon peak hours, while others may not), Yang et al. [14] proposed Spatial-Temporal Hidden Markov models (STHMM). The dependencies and the correlations among different time series were modeled while considering the topology of the road network. Wang et al. [6] modeled different drivers' travel times with a three-dimension tensor, the frequent trajectory patterns were extracted from historical tips to decrease the candidates of concatenation and suffix-tree-based indexes, then an object function was devised and proved to model the tradeoff between the length of a path and the number of trajectories traversing the path. The object function was then solved by a dynamic programming solution. Wen et al. [8] proposed a novel probability-based method by constructing a temporally weighted spatial-temporal distribution patterns to estimate the logistical transport time. In order to explore location and time relationship, they designed frequent spatial connections, in which area-based spatial-temporal probabilistic distribution can be identified by kernel density estimation. Then the transportation time between two locations in the area can be estimated. Similarly, Jabari et al. [15] established a mixture asymmetric probabilistic statistical framework, i.e., a novel data-driven methodology of Gamma mixture densities, to model complexity multi-modal urban travel time distributions, experiments also demonstrated their methods can further solve the data sparsity.

### 2.2 Deep Learning Travel Time Prediction Approaches

To cope with the insufficiency of the input information, Li et al. [11] constructed a more smooth and meaningful multi-task representation learning by leveraging the underlying road network structure and spatial-temporal prior knowledge. Jindal et al. [13] first predicted the distance between the origin and destination, and then estimated the travel time based on the above predicted distance. The advance of ST-NN was that it only take advantage of the raw trips data without demanding further feature engineering. However, the road network structure, i.e., the spatial and temporal relationship, was neglected. In order to solve the inaccuracies caused by the divide-and-conquer methods, Wang et al. [16] proposed a novel end-to-end deep learning framework to estimates the travel time of the whole path directly, they used a geo-convolutional and LSTM layer to capture the spatial and temporal features meantime.

In addition, they also introduced a multi-task learning to balance the effect of the entire path and each local path. However, they didn't consider the influences brought by the driver's habit and external road types. Zhang et al. [17] proposed an end-to-end training-based model named DEEPTRAVE to predict the travel time of a whole path directly, and designed an auxiliary supervision with dual interval loss mechanism to fully leverage the temporal existing historical labeling information. They utilized a feature extraction structure to effectively capture different dynamics, such as short-term and long-term traffic features, for estimating the travel time accurately. Cui et al. [18] presented a deep stacked bidirectional and unidirectional LSTM (SBULSTM) neural network architecture, which investigated both spatial features and bidirectional temporal dependencies from historical data. This mechanism can effectively handle the missing value for input data, and can also address the passing information from a back-propagation direction.

### 3 Methodology

In this section, we formally depict the preliminaries and define the notions of the problem, then present the details of our method.

#### 3.1 Preliminaries

**Definition 1.** A road network  $G = (V, E)$  is a directed graph, the  $V = \{v_i(x_i, y_i)\}$  represents a set of vertices, each  $v_i$  incorporates latitude  $x_i$  and longitude  $y_i$ , the  $E = \{e_j(v_m, v_n)\}$  represents a set of edges, each  $e_j$  is comprised of two directly connecting vertices.

**Definition 2.** A trajectory  $T$  is a sequence of GPS points generated from LBS (Location Based Service) devices, which can be denoted as  $T = \{p_1, p_2, \dots, p_n\}$ . Each GPS point  $p_i$  contains 5-tuple  $(t_{pi}, x_i, y_i, velocity_i, head_i)$ , where  $t_{pi}$  denotes the timestamp,  $x_i$  and  $y_i$  denote the latitude and longitude respectively,  $velocity_i$  is the vehicle speed,  $head_i$  is the angle of driving direction.

**Problem:** Given a trajectory  $T$  and departure time  $t_o$ , our goal is to estimate the travel time for  $T$  by using a series of historical trajectories on road network  $G$ .

#### 3.2 Framework of STDR

As presented in Fig. 1, we provide the details for our proposed spatial-temporal network framework, which is comprised of three major components.

**Spatial Component:** We first leverage the road network embedding method to embed the GPS points into corresponding vectors, thus we can use the convolutional layer with many filters to extract the spatial characteristic [9], after that the output matrix vectors of convolutional operation are used as the input of LSTM.

**Temporal Component:** We manually collect some external features from external datasets, such as time of day, weather conditions, etc., and then embed them as a vector

$X_{ext}$ , and concentrate with vectors  $C_{IN}$  which are obtained from above spatial component, moreover feed them into two stacked LSTM with attention mechanism to extract the temporal characteristic with attention mechanism.

**Road Type Component:** We introduce a segmentation approach which is capable of dividing the trajectory into sub-segments based on the road types, then calculate each sub-segments average distance, which is further encoded into vectors. To enable the full connected operations on the irregular vectors matrix, we pad zero where it is necessary. Then the vectors further combine the output of the LSTM component, and furthermore go through a fully connected layer at the end of the network for joint prediction.

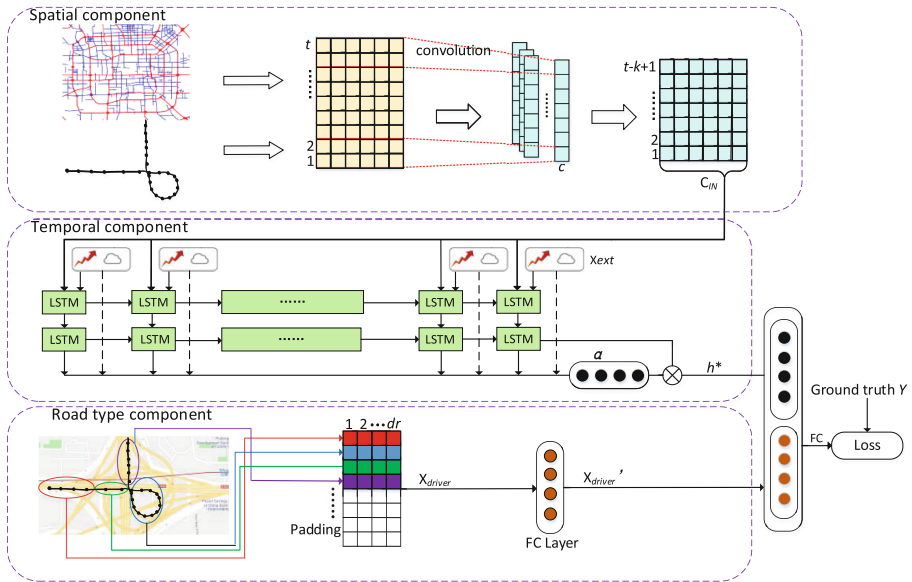


Fig. 1. The framework of STDR

### 3.3 Trajectory Embedding Representation

In [16], the geographical features they extracted are multiplied by latitude and longitude of two points, which don't consider the global road network dependencies among different vertices. Therefore, we first need to convert the GPS points into vectors rather than working on them directly [9]. How to transfer the trajectory can be recognized as the problem of embedding very large road networks into low-dimensional vector spaces, aiming to capture and preserve the original network structure. The characteristics of vertices are dependent on both the local and global network structure. Therefore, how to

simultaneously preserve above structures is a tough problem. In this paper, we use a road network embedding method [19], which can effectively handle the distance and angle in the road network space by using the Minkowski  $p$ th-order metrics:

$$L_p(x, y) = \left[ \sum_{i=1}^k |x_i - y_i|^p \right]^{1/p} \tag{1}$$

Transforming a road network  $G = (V, E)$  into a vector space  $(R^L, D')$  is a mapping  $V \rightarrow R^L, E \rightarrow D'$ , where  $L$  is the vectors' dimension and  $D'$  is one of Minkowski metrics [19, 22]. The vertex is extended as follows: Let  $V_i$  be a subset of  $V$ ,  $x$  be a point, and  $Dist(x, V_i) = \min_{y \in V_i} \{Dist(x, y)\}$ , here  $Dist(x, V_i)$  is the distance from point  $x$  to its closet neighbor in  $V_i$ . Then let set  $R = \{V_{1,1}, \dots, V_{1,k}, \dots, V_{\beta,1}, \dots, V_{\beta,k}\}$  be a subset of  $V$ , where  $k$  is set as  $O(\log n)$  and  $\beta$  is set as  $O(\log n)$ , the original space can be embedded into a  $O(\log^2 n)$  dimensional space. Specially, let  $E(v) = (R_{V_{1,1}}(v), \dots, R_{V_{1,k}}(v), \dots, R_{V_{\beta,1}}(v), \dots, R_{V_{\beta,k}}(v))$ , in which  $R_{V_{i,j}}(v) = Dist(v, V_{i,j})$ , thus for single point  $v$ , the finally output is vector  $E(v) \in R^{\beta * k}$  with the dimension  $\beta * k$  (the following is denoted by  $L = \beta * k$  for the rest paper) on the road network. For dynamic insertion adjustment in one specified reference subset  $V_{a,b}$ , the new vertex  $V_N$  is modified by  $R_{V_{a,b}}(V_N) = \min(Dist(V_N, P_i) + Dist(P_i, V_{a,b}), Dist(V_N, P_j) + Dist(P_j, V_{a,b}))$ , where  $Dist(P_i, V_{a,b})$  is distance between  $P_i$  and  $V_{a,b}$ , as presented in Fig. 2.

After the trajectory embedding, we can obtain  $t$  vectors  $\in R^L$ , where  $t$  is the number of trajectory points.

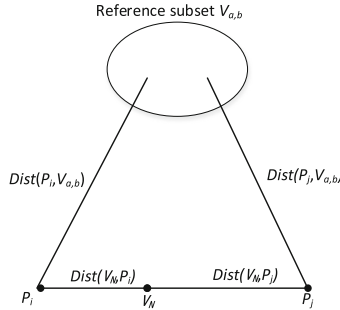


Fig. 2. Dynamic embedding for trajectory points

### 3.4 Trajectory Spatial Characteristics Captured by Convolutional Layer

Inspired by the successful applications of CNN on images, we also employ it to extract the GPS spatial characteristics. Compared with image having two-dimensions spatial structure, trajectory has only one-dimensional spatial structure like the sentence in word2vec classification model, where words display in sequence. Thus we use the one-dimension convolutions method to learn pixel-level spatial correlation features by considering the trajectory points as an image of width  $t$  and height 1.

As in Fig. 1, let  $x_i \in \mathbb{R}^L$  [9] be the  $L$  dimensional road network embedding vector corresponding to the  $i$ -th point in the trajectory sequence, the trajectory with length  $t$  is presented as:

$$x_{1:t} = x_1 \oplus x_2 \oplus \dots \oplus x_t \quad (2)$$

where  $\oplus$  thus the trajectory can be converted as a vector matrix, with the size  $L * t$ . A convolution operation involves a filter  $w \in \mathbb{R}^{L*k}$ , whose kernel window size is  $k$ , and is overlaid across the GPS points vectors ranging from  $i$  to  $i + k - 1$ . Next, it performs an element-wise product, and then adds them together and obtains one new feature. For example, the transformation from a window of  $x_{i:i+k-1}$  is defined as follows:

$$c_i = \delta(w_i \cdot x_{i:i+k-1} + b) \quad (3)$$

where  $w_i$  is a weight parameter and  $b$  is a bias and  $\delta(\cdot)$  is a non-linear function. Thus, a feature vector is generated from one filter, which is successively applied to GPS points  $(x_{1:k}, x_{2:k+1} \dots x_{t-k+1:t})$  where stride equals 1, with the index ranging from 1 to  $t-k+1$ . Finally, for each trajectory connected by  $c$  filters, we get the output vectors  $C_{IN} \in \mathbb{R}^{c*(t-k+1)}$  presented by Formula 4.

$$C_{IN} = [c_1, c_2, \dots, c_{(t-k+1)}] \quad (4)$$

### 3.5 Trajectory Temporal Characteristics Captured by LSTM

A successful approach for solve the time sequential problems is RNN (Recurrent Neural Network), which can remember the previous historical sequence by using a transition function and leveraging an internal memory to process the dynamic temporal behavior. RNN has proven the ability to model variable length sequence. However, traditional RNN may also face the gradients exploding or vanishing because the time sequences is too long, thereby the LSTM is designed [20] since it can decide whether or not to abandon the previous hidden states depending on the time restrictions. Generally speaking, LSTM extends RNN by adding three gate (i.e., one input gate, one forget gate, one output gate) and a memory cell. The forget gate is employed to abandon some irrelevant information and can effectively solve the vanishing or exploding gradient problem. The input gate and output gate are utilized to control the input and output vectors. The output is the last hidden state of LSTM. At each time interval  $t$ , LSTM takes the output of convolutional layer as an input, and then all information is accumulated to the memory cell, each cell in LSTM is defined as follows:

$$f_t = \delta(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (5)$$

$$i_t = \delta(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (6)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (7)$$

$$C_t = f_t \circ C_{t-1} + i_t \circ \tilde{C}_t \quad (8)$$

$$o_t = \delta(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (9)$$

$$h_t = o_t \circ \tanh(C_t) \quad (10)$$

where  $f_t, i_t, o_t$  represent the forget gate, input gate, and output gate,  $W_f, W_i, W_c, W_o$  are the weight parameters matrices,  $b_f, b_i, b_c, b_o$  are the biases values,  $\delta(\cdot)$  denotes the non-linear activation function, the  $\tanh$  denotes the hyperbolic tangent function. Furthermore, the multi-layers LSTM is more efficient than a single LSTM layer [20].

Intuitively, the travel time can be affected by many complex external factors, such as weather conditions and time metadata (i.e., time-of-hour, day-of-week). For instance, traffic on rainy days is usually more congested than usual, and the road is more prone to have high-level crowd [3], etc. Note that these external factors cannot be directly fed into a neural network, we embed the weather conditions [4] as  $X \in \mathbb{R}^{16}$ , time-of-hour as  $X \in \mathbb{R}^{24}$ , day-of-week as  $X \in \mathbb{R}^7$ , etc., by using the hot coding. Then the individual vectors are concatenated with the output of CNN, and fed into LSTM units.

**Attention Mechanism Model:** The traditional LSTM cannot detect which segment has a greater weight for estimating the whole trajectory time. For example, the impacts of expressway and speed limited road on the whole estimated time are different from each other. In order to address this issue, we design an attention mechanism [21] that can describe the key part of segments among whole trajectory segments. Let  $X_{ext} \in \mathbb{R}^{e \times (t-k+1)}$  represents the embedding of external factors, we append it to  $C_{IN}$  as LSTM input presented by Formula (11). Furthermore, Let  $H \in \mathbb{R}^{c \times (t-k+1)}$  be a matrix comprised of hidden vectors  $[h_1, h_2, \dots, h_{t-k+1}]$  that the LSTM produced in Formula (10). As in Figs. 1 and 3, the attention vector takes hidden vectors  $H$  and  $X_{ext}$  as input to compute the probability distribution of source trajectory input. By utilizing this mechanism, it is possible for finally prediction to capture somewhat global information rather than solely to derive from hidden states. A vector including attention weights and a weighted hidden representation of GPS points are denoted as  $\alpha, h^*$  respectively. The details are presented from Formula 12 to 14.

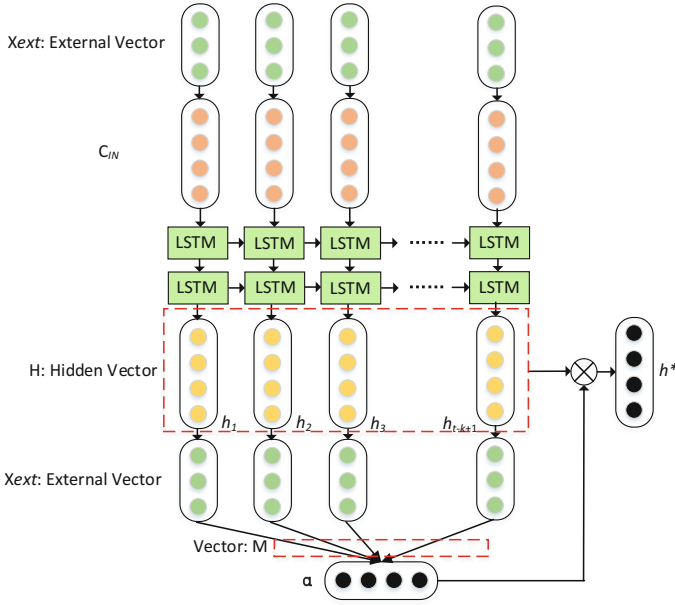
$$C_{IN} = W_{in}C_{IN} + W_{ext1}X_{ext} \quad (11)$$

$$M = \tanh\left(\begin{bmatrix} W_h H \\ W_{ext2} X_{ext} \end{bmatrix}\right) \quad (12)$$

$$\alpha = \text{softmax}(\omega^T M) \quad (13)$$

$$h^* = H\alpha \quad (14)$$





**Fig. 3.** The architecture of LSTM attention mechanism

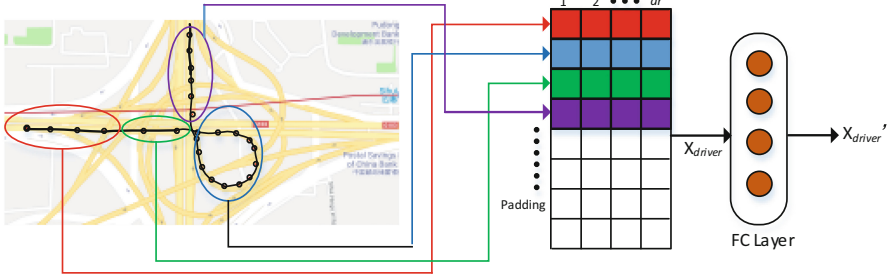
where  $W_{in} \in \mathbb{R}^{c \times c}$ ,  $W_{ext1} \in \mathbb{R}^{c \times e}$ ,  $W_h \in \mathbb{R}^{c \times c}$ ,  $W_{ext2} \in \mathbb{R}^{e \times e}$ ,  $\omega^T \in \mathbb{R}^{1 \times (c+e)}$ ,  $M \in \mathbb{R}^{(c+e) \times (t-k+1)}$ ,  $\alpha \in \mathbb{R}^{(t-k+1)}$ ,  $h^* \in \mathbb{R}^c$ . Finally, the output of the temporal component is  $h^*$ .

### 3.6 The Detail of Road Types Embedding

In fact, different road types have different effects on the travel time. For example, driving on the overpass is more time-consuming than on the highway at the same time interval. As in Fig. 4, a sample trajectory passes on the auxiliary road, urban expressway road, overpass road, urban expressway road sequentially, different parts of the trajectory exhibit diverse driving speeds marked out by ovals. For instance, the blue oval denotes the overpass road, whose sampling locations are close to each other, while the red oval denotes urban expressway road, whose sampling locations are far away from each other.

In summary, we propose a trajectory segmentation approach [22, 23]. Trajectory segmentation is a fundamental task which tries to partition a trajectory into several segments based on a set of optimization goals. We aim to find the characteristic points where the shape of a trajectory changes rapidly. The segmentation includes two desirable properties: preciseness and conciseness [23], we leverage the concept of Minimal Description Language (MDL) to find the optimal tradeoff between preciseness and conciseness.

The MDL is comprised of two components:  $L(H)$  and  $L(D|H)$ .  $L(H)$  is regarded as the hypothesis description with the length of the data described in bit, which measures



**Fig. 4.** The framework of road type component (Color figure online)

the degree of conciseness, while the  $L(D|H)$  is regarded as the length of the description of data under the hypothesis  $H$ , which measures preciseness. In our paper,  $L(H)$  is simply equal to  $\log_2 x$ . Furthermore, the  $L(H)$  represents the total length of the Euclidean distance between all  $p_j$  and  $p_{j+1}$ , while the  $L(D|H)$  represents the sum of the difference between a trajectory and its trajectory partitions. At last, a list of characteristic points is picked out by minimizing  $L(H) + L(D|H)$ . The trajectory is partitioned into segments by these characteristic points.

Then we obtain the average distance of every segment, and embed them as vector  $X_{driver} \in \mathbb{R}^{8*dr}$  groups padding whatever it is necessary. Finally, the  $X_{driver}$  is connected with FC layer to yield vector  $X_{driver'} \in \mathbb{R}^{dr}$ , which concatenates LSMT to jointly train the model, thus the Formula 14 is rewritten as:

$$h^* = W_{h^*}h^* + W_{driver}X_{driver'} \quad (15)$$

where  $W_{h^*} \in \mathbb{R}^{c*c}$ ,  $W_{driver} \in \mathbb{R}^{c*dr}$ .

### 3.7 Prediction Component

The next step is to estimate the travel time by integrating the output of LSTM and the output of road type component. We feed the  $h^*$  into the fully connected network to get the final estimated value  $\tilde{Y}_t$ , which is calculated as follows:

$$\tilde{Y}_t = \tanh(W_{of} \cdot h^* + b_{of}) \quad (16)$$

where  $W_{of}$  and  $b_{of}$  are learnable parameters,  $\tanh(x)$  is a hyperbolic tangent function, which ensures the output values are in  $[-1 \sim 1]$ . The loss function we used is defined by minimizing the mean squared error between the estimated time and the true time:

$$L(\theta) = |Y_t - \tilde{Y}_t|^2 \quad (17)$$

where  $\theta$  is the set of all learnable parameters needed to be trained. We continuously adjust the parameter sets using back propagate by Tensorflow until loss function converges.

## 4 Experiments and Discussions

### 4.1 Dataset

**Datasets:** We test the method on the Beijing road network including about 330,000 vertices and 440,000 edges. We use two GPS trajectory datasets named **Taxi** and **Ucar** [24]. The **Taxi** data contains about 180,000 trajectories generated by more than 7,000 public taxis, **Ucar** data contains about 480,000 trajectories generated by more than 6,000 private taxis in November 2015. Each sampling point includes timestamp, latitude, longitude, vehicle speed, and direction. The abnormal records are first filtered out, and then the map matching algorithm is employed to relocate the deviated sample points. The data is divided into two subsets: we use the first 24 days data as the training data, the rest days as the test data.

**Meteorological data:** We record the Beijing weather data from Beijing Meteorological Bureau, the data include rainfall, temperature, wind velocity and so on. The weather conditions are divided into 14 types: cloudy, sunny, heavy rain, middle rain, light rain, heavy snow, light snow, dense fog, little fog, overcast, hail, frost, smog and haze, and sand storm.

### 4.2 Parameters Setting

Our model is implemented with Python 2.0. The model is deployed on the server with Core i7-4790 CPU, 16 GB RAM, NVIDIA GTX1080 GPU. We adopt Adam optimization algorithm with mini-batch size equals to 512 to train the parameters, the initial learning rate is set as 0.01.

### 4.3 Baseline Algorithms for Comparison

To demonstrate the validity of our model, we compare it with 5 baseline methods including:

**ARIMA:** ARIMA means Auto Regressive Integrated Moving Average, which is a typical and well known statistical model that depicts a suite of different standard temporal attributions.

**XGBoost:** XGBoost is an efficient, flexible machine learning technique for regression, classification and sorting tasks by assembling multiple weak learning under the gradient boosting framework, usually referring to decision trees. It belongs to ensemble learning.

**SimpleTTE [7]:** SimpleTTE presents a Simple Travel Time Estimating method that leverages the neighboring trips from the large amount of historical data. Being different from the traditional approaches, SimpleTTE indexes all the neighboring points with similar original and destination, and calculates the absolute temporal speed reference under irregularities traffic condition. After that, the travel time is scaled and estimated from the similar trips with the original demands.

**Multi MASK [11]:** Multi MASK is a multitasking representation learning model for time estimation, which does not hypothesize that the travel route is predetermined, but

utilizes the underlying road network with time and space prior knowledge. This method also engenders a meaningful representation that retains the various travel attributes.

DeepTTE [16]: DeepTTE is an end-to-end deep learning framework approach for estimating travel time of the whole path directly, the method presents a geo-convolution operation to capture the spatial correlations, and leverages stacking LSTM layers to capture the temporal dependencies as well. In addition, the relationships between the local and the global tradeoff are determined by the multitasking they presented. We apply this algorithm following the parameter settings deployed in [16].

#### 4.4 Evaluation Metrics

We evaluate the performance of the proposed method based on three popular metrics. Assume  $y_1, y_2, \dots, y_n$  denote the ground truth,  $\tilde{y}_1, \tilde{y}_2, \dots, \tilde{y}_n$  denote the estimated value, and  $n$  denotes the numbers of samples points. Here, Mean Absolute Percentage Error (MAPE), Mean Relative Error (MRE), and Mean Absolute Error (MAE) are employed as evaluation metrics, their definitions are as follows:

$$MAPE = \sum_{t=1}^n \left| \frac{y_t - \tilde{y}_t}{y_t} \right| * \frac{1}{n} \quad (18)$$

$$MRE = \frac{\sum_{t=1}^n |y_t - \tilde{y}_t|}{\sum_{t=1}^n |y_t|} \quad (19)$$

$$MAE = \frac{\sum_{t=1}^n |y_t - \tilde{y}_t|}{n} \quad (20)$$

#### 4.5 Performance Comparisons

Table 1 shows the comparisons between baseline algorithms and our presented method.

**Table 1.** Performance comparisons with baselines

Model	Taxi			Ucar		
	MAPE(%)	MRE(%)	MAE(s)	MAPE(%)	MRE(%)	MAE(s)
ARIMA	35.49	32.18	257	32.32	30.1	243
XGBoost	34.45	31.18	234	32.99	28.2	227
SimpleTTE	26.93	25.71	213	22.75	22.3	219
Multi MASK	23.35	22.63	207	21.53	19.45	186
DeepTTE	19.37	18.52	191	17.61	16.8	164
Ours STDR	<b>16.19</b>	<b>15.54</b>	<b>155</b>	<b>15.04</b>	<b>13.31</b>	<b>136</b>

From the Table 1, we can see that the MAPE, RMSE, and MAE of the ARIMA and XGboost perform poor results, which demonstrates that the simple traditional prediction method cannot effectively describe the large scale complex spatial-temporal data. The DeepTTE and Multi MASK both outperform ARIMA and XGBoost, the comparisons reveal that deep learning methods can work well. Furthermore, since the DeepTTP considers utilizing the convolutional operation to capture the spatial characteristic, it shows the better result than Multi Mask. Next, it is interesting the SimpleTTE method displays an accept medium performance between the traditional and deep learning methods. However, it is just an approximate method, which is more fit for the ideal situation, such as the highway or urban expressway with little speed changing. Finally, our algorithm significantly outperforms above mentioned methods with the lowest MAPE (16.19% and 15.04%), MRE (15.54% and 13.31%), and MAE (155 and 136) on two datasets respectively, which verifies the superiority and feasibility of our approach. The reason is that our algorithm further exploits LSTM attention mechanism and takes account of the influence of road type in the whole trajectory, these settings can better preserve the spatial-temporal characteristics of the original trajectory.

#### 4.6 Comparison with Different Variants

To investigate the effectiveness of different components in Fig. 1, we compare our STDR with 4 different varieties including: (1) LSTM and road type without CNN component. (2) Only LSTM component, neither the CNN nor the road type is used. (3) CNN and LSTM components without road type. (4) LSTM without attention mechanism. All these models have identical inputs and the parameters are roughly the same. The results are presented in Table 2.

**Table 2.** Evaluation of our method and its variants

Model	Taxi			Ucar		
	MAPE (%)	MRE (%)	MAE (s)	MAPE (%)	MRE (%)	MAE (s)
①Only LSTM component	32.14	31.82	234	29.47	26.53	225
②LSTM + road type without CNN	29.25	26.72	207	25.78	24.38	183
③CNN + LSTM without road type	22.48	21.68	188	22.16	21.63	167
④STDR without attention	20.31	17.56	171	18.86	15.74	158
Our STDR	<b>16.19</b>	<b>15.54</b>	<b>155</b>	<b>15.04</b>	<b>13.31</b>	<b>136</b>

From Table 2, we have the following observations. Firstly, it is not surprising that only LSTM component exhibits the lowest performance as expected. Secondly, considering the ②LSTM + road type without CNN and the ③CNN + LSTM without road type, the impacts of CNN (MPAE is 22.48%) is more obvious than the road type (MAPE is 29.25%). The reason for this phenomenon is that some road type

characteristics can be also extracted by CNN in spatial component. Thirdly, both two variants (i.e., the ③CNN + LSTM without road type and ④STDR without attention) outperform the two variants without CNN (i.e., the ②LSTM + road type without CNN, and the ①only LSTM component), which demonstrates the significant role of CNN in spatial trajectory data mining. Fourthly, the ④STDR without attention is weaker than our STDR, the explanation is that the error of various types roads will continuously accumulate as the trajectory sequence grows, which confirms the effectiveness of attention mechanism. Finally, the performance of our STDR achieves the best when all aspects are considered.

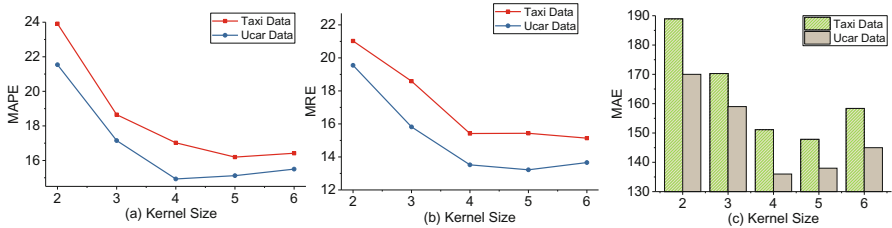


Fig. 5. Impacts of kernel size

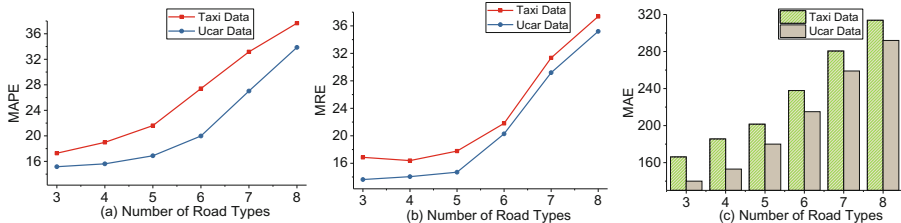


Fig. 6. Impacts of the number of road types

#### 4.7 Impacts of Kernel Size and the Number of Road Types

We first evaluate the performance of kernel size of the convolutional filter. From Fig. 5, we observe that the MAPE of both data decreases as the kernel size grows, this discloses that the large kernel size can better capture the far away spatial dependences on the trajectory. However, when the kernel size exceeds 4, the effect becomes doubtful and the MAPE even rises. The cause is that although a larger filter can capture more information, it also imports much unnecessary noise, damaging the original road network correlation, such as two contradictory features generated by two successive turnings.

The impacts of the number of road types are exhibited in Fig. 6. As we can see, the results reveal that when the number of road type is smaller than 6, the corresponding average distance on the road is about 14 km according to historical trajectories statistics, the MAPE, MRE, and MAE are nearly unchanged, the reason may be related

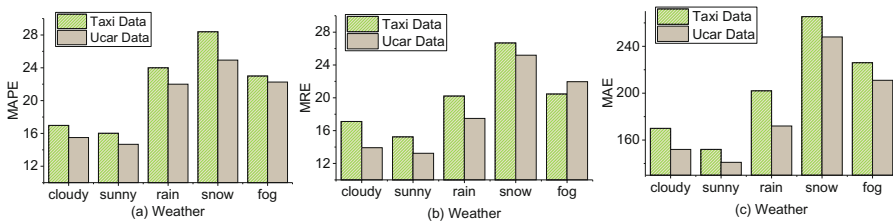
to that the Beijing’s layout of streets is in the shape of regular block layout. However, as the number of road types increases continually, the growth ratio rises dramatically, this indicates that more influencing factors produced by a long trajectory on the road can easily result in inaccurate predictions.

### 4.8 Impact of Weather Conditions

In this paper, we estimate the travel time by utilizing the weather forecasting as the approximate weather at future time interval  $n + 1$ . But in fact, weather forecasting is not fairly accurate all the time due to the technology and so on. To investigate the effectiveness of the weather component, first we remove it and compare it with STDR, then pick out five typical kinds weather, i.e., cloudy, sunny, rain, snow, and fog. The comparisons are shown in Table 3 and Fig. 7.

**Table 3.** Experimental results without weather conditions

Model	Taxi			Ucar		
	MAPE (%)	MRE (%)	MAE (s)	MAPE (%)	MRE (%)	MAE (s)
STDR without weather conditions	23.71	19.48	196	21.53	16.21	177
<b>Our STDR</b>	<b>16.19</b>	<b>15.54</b>	<b>155</b>	<b>15.04</b>	<b>13.31</b>	<b>136</b>



**Fig. 7.** Impact of weather conditions

In Table 3, we can see that the performance of STDR without weather conditions decrease by 31.7% comparing with our STDR (w.r.t. Taxi’s MAPE), this indicates the weather has a significant influence on travel time. Next, from Fig. 7, we can discover that the MAPE on cloudy and sunny days are almost the same, suggesting that good weather condition has few impacts on travel time. On the rainy and foggy days, the results are relatively poor, and are generally worse than cloudy and sunny days, this is because bad weather can affect people’s attention and result in slow response. The outcome on snowy days is the worst, the cause is that drivers need to be much longer waiting time at intersections due to the wet slippery road, this also conforms to our intuitive sense.

## 5 Conclusions

In this paper, we propose a novel deep learning end-to-end model based on CNN and LSTM for estimating travel time by using real historical traffic data. The method first embeds trajectory into low dimension vectors with the road network, then employs CNN to capture the spatial characteristic, further utilizes LSTM with attention mechanism to capture the time sequence characteristic. What's more, we import the road segmentation to fully depict the influence of road type. To validate the effectiveness of the proposed STDR, extensive experiments with 5 baselines are conducted. The results, in terms of MAPE, MRE, and MAE, demonstrate the superiority of our methodologies. In the future we plan to work on three interesting directions: (1) Incorporate the social network for the estimating travel time model. (2) Apply machine learning to interdisciplinary areas such as smart transportation and economics disciplines. (3) Extend and apply the framework to other trajectory problems.

**Acknowledgements.** This work was supported by NSFC(91646202), National Key R&D Program of China (SQ2018YFB140235), and the 1000-Talent program.

## References

1. Yi, X., Zhang, J., Wang, Z., Li, T., Zheng, Y.: Deep distributed fusion network for air quality prediction. In: SIGKDD, pp. 965–973 (2018)
2. Zhang, S., Wu, G., Costeira, J.P., Moura, J.M.: FCN-rLSTM: deep spatio-temporal neural networks for vehicle counting in city cameras. In: ICCV, pp. 3687–3696 (2017)
3. Zhang, J., Zheng, Y., Qi, D.: Deep spatio-temporal residual networks for citywide crowd flows prediction. In: AAAI, pp. 1655–1661 (2017)
4. Yao, H., et al.: Deep multi-view spatial-temporal network for taxi demand prediction. In: AAAI, pp. 2588–2595 (2018)
5. Amirian, P., Basiri, A., Morley, J.: Predictive analytics for enhancing travel time estimation in navigation apps of Apple, Google, and Microsoft. In: SIGSPATIAL, pp. 31–36 (2016)
6. Wang, Y., Zheng, Y., Xue, Y.: Travel time estimation of a path using sparse trajectories. In: SIGKDD, pp. 25–34 (2014)
7. Wang, H., Kuo, Y. H., Kifer, D., Li, Z.: A simple baseline for travel time estimation using large-scale trip data. In: SIGSPATIAL, pp. 61:1–61:4 (2016)
8. Wen, R., Yan, W., Zhang, A.N.: Adaptive spatio-temporal mining for route planning and travel time estimation. In: Big Data, pp. 3278–3284 (2017)
9. Kim, Y.: Convolutional neural networks for sentence classification. In: EMNLP, pp. 1746–1751 (2014)
10. He, K., Zhang, X., Ren, S., et al.: Deep residual learning for image recognition. In: CVPR, pp. 770–778 (2016)
11. Li, Y., Fu, K., Wang, Z., Shahabi, C., Ye, J., Liu, Y.: Multi-task representation learning for travel time estimation. In: KDD, pp. 1695–1704 (2018)
12. Wang, J., Gu, Q., Wu, J., Liu, G., Xiong, Z.: Traffic speed prediction and congestion source exploration: a deep learning method. In: ICDM, pp. 499–508 (2016)
13. Jindal, I., Chen, X., Nokleby, M., Ye, J.: A unified neural network approach for estimating travel time and distance for a taxi trip. In: CoRR, abs/1710.04350 (2017)



14. Yang, B., Guo, C., Jensen, C.S.: Travel cost inference from sparse, spatio temporally correlated time series using Markov models. In: VLDB, pp. 769–780 (2013)
15. Jabari, S.E., Freris, N.M., Dilip, D.M.: Sparse travel time estimation from streaming data. In: CoRR, abs/1804.08130 (2018)
16. Wang, D., Zhang, J., Cao, W., Li, J., Zheng, Y.: When will you arrive? Estimating travel time based on deep neural networks. In: AAAI, pp. 2500–2507 (2018)
17. Zhang, H., Wu, H., Sun, W., Zheng, B.: DeepTravel: a neural network based travel time estimation model with auxiliary supervision. In: IJCAI, pp. 3655–3661 (2018)
18. Cui, Z., Ke, R., Wang, Y.: Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction. In: arXiv, abs/1801.02143 (2018)
19. Shahabi, C., Kolahdouzan, M.R., Sharifzadeh, M.: A road network embedding technique for k-nearest neighbor search in moving object databases. *GeoInformatica* **7**(3), 255–273 (2003)
20. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.* **9**(8), 1735–1780 (1997)
21. Wang, Y., Huang, M., Zhao, L.: Attention-based LSTM for aspect-level sentiment classification. In: EMNLP, pp. 606–615 (2016)
22. Zheng, Y.: Trajectory data mining: an overview. *ACM TIST* **6**(3), 29:1–29:41 (2015)
23. Lee, J.G., Han, J., Whang, K.Y.: Trajectory clustering: a partition-and-group framework. In: SIGMOD, pp. 593–604 (2007)
24. Ta, N., Li, G., Zhao, T., Feng, J., Ma, H., Gong, Z.: An efficient ride-sharing framework for maximizing shared route. *TKDE* **30**(2), 219–233 (2018)



# Using Fractional Latent Topic to Enhance Recurrent Neural Network in Text Similarity Modeling

Yang Song<sup>(✉)</sup>, Wenxin Hu, and Liang He

Department of Computer Science and Technology,  
East China Normal University, Shanghai 200241, China  
ysong@ica.stc.sh.cn, wxhu@cc.ecnu.edu.cn, lhe@cs.ecnu.edu.cn

**Abstract.** Recurrent neural networks (RNNs) have been widely used in text similarity modeling for text semantic representation learning. However, referring to the classical topic models, a text contains many different latent topics, and the complete semantic information of the text is described by all the latent topics. Previous RNN based models usually learn the text representation with the separated words in the text instead of topics, which will bring noises and loss hierarchical structure information for text representation. In this paper, we proposed a novel fractional latent topic based RNN (FraLT-RNN) model, which focuses on the text representation in topic-level and largely preserve the whole semantic information of a text. To be specific, we first adopt the fractional calculus to generate latent topics for a text with the hidden states learned by a RNN model. Then, we propose a topic-wise attention gating mechanism and embed it into our model to generate the topic-level attentive vector for each topic. Finally, we reward the topic perspective with the topic-level attention for text representation. Experiments on four benchmark datasets, namely TREC-QA and WikiQA for answer selection, MSRP for paraphrase identification, and MultiNLI for textual entailment, show the great advantages of our proposed model.

**Keywords:** Latent topic · Fractional calculus · Recurrent neural network

## 1 Introduction

Text similarity modeling is a crucial issue in many neural language processing (NLP) tasks, such as paraphrase identification [4, 10], question answering [25, 35], and textual entailment [18, 23]. Take the paraphrase identification task as an example, text similarity is utilized to assess whether the two pieces of texts are semantically equivalent.

Recently, the recurrent neural networks (RNNs) have gained popularity in text similarity modeling, due to its good performance and less human interventions. Specifically, a hidden vector is learned for each word in the text via a hidden state in

RNN, and the whole text is represented by the aggregation of all the hidden vectors. Then, the similarity of a pair of texts is calculated with their representations and a similarity function. Most RNN based models, including those embedded with attention mechanisms, focus on using sequential hidden vector in word-level to generate the text representation, while the hierarchical structures of the text, such as features in phrase-level and sentence-level, are neglected. However, a piece of text has complicated structures, it is essential to understand and represent the text both sequentially and hierarchically [15].

Referring to the well-known topic models, such as latent dirichlet allocation (LDA) [3], it can be found that words in the text generate various topics, and the distribution of the topics demonstrates the semantic representation of the text. Noting that a topic is generated by a group of words, [4, 35] use the aligned textual information to model text similarity, which first locates the textual snippets that have the same semantic meanings in the text pair, and then highlights the weight of those textual snippets for text representation. However, this method usually focuses on the co-occurrent words between the text pair, instead of the high-level topics, which will bring noise during text similarity modeling. Take the following text pair as an example for illustration.

*T1: A child gets a fever, but he has no symptoms of influenza.*

*T2: A kid with the symptoms of mild influenza and low fever can be cured by the Oseltamivir.*

Obviously, there are two topics in T1, namely “the child has a fever” as topic 1, and “the child has no influenza” as topic 2. It can be seen that topic 1 is relevant to T2, while topic 2 is not. Therefore, T1 and T2 should have a lower similarity. However, textual alignment approaches pay more attention on the aligned words, such as “fever” and “influenza”, and will generate a higher similarity score for T1 and T2 and lead to mismatching problem. Hence, modeling the text similarity in topic-level is meaningful and promising, which should arouse much attention.

To the best of our knowledge, how to generate topics based on the words interactions and use interactions among topics for text similarity modeling are still not well studied in RNN. In this paper, we propose a **fractional latent topic** based RNN (**FraLT-RNN**) model, where the hierarchical features, namely features in word-level and topic-level, as well as the word sequential patterns, are incorporated into RNN for text representation by means of the fractional latent topics. To be specific, we first adopt the fractional latent topic generator to learn latent topics based on the hidden states learned by a RNN structure. In particular, the fractional latent topic generator is derived from the fractional calculus, which computes the function’s integral in fractional order, instead of integer order, and has been successfully introduced into image processing for generation and denoising, due to its excellent characteristics in memory and heredity. Then, we design a topic-wise attention mechanism to generate an topic-level attentive vector for each latent topic, which measures the perspective of the latent topic and enhances the interactions between a text pair. Finally, the latent topics are rewarded by the attentive vector for text representation and similarity calculation. We evaluate

our FraLT-RNN model on four benchmark collections, namely Trec-QA and WikiQA for question answering, MSRP for paraphrase identification, and MultiNLI for textual entailment. The experimental results show great advantages of the proposed FraLT-RNN on text similarity modeling. It is notable that we achieve the new state-of-the-art performance on TREC-QA, WikiQA, and MSRP. Furthermore, our model is comparable to if not better than the recent neural network based approaches on MultiNLI.

The contribution of this paper is summarized as follows:

- We propose a new fractional latent topic based RNN model, where the text is represented in topic-level for better semantic capturing and understanding.
- This is the first attempt to introduce the fractional calculus into neural language processing for latent topics generation.
- We conduct elaborate analyses of the experimental results on three text similarity tasks, which provides a better understanding of the effectiveness of our model.

## 2 Related Work

Recently, the deep neural networks have been widely used in text similarity modeling [26,32], especially the recurrent neural networks (RNN) due to their capacity in modeling the sentence with variable length. [7,42] applied the long short-term memory (LSTM) [12] based RNN model to obtain the semantic relevance between text pairs for the community based question selection.

To capture the salient information for better sentence representations, the attention mechanism was introduced into the neural networks [25,31]. [41] proposed an attentive interactive neural network, which focused on the interactions between text segments for answer selection. In addition, the interactions in sentence-level or word-level are incorporated for the attentive weight generation within the RNN framework. In [29], the attentive weights for an answer sentence relied on the interactions with the question sentence. In [25], the word-by-word interactions were utilized for the attentive sentence representations.

Most of the previous work focused on representing the text in word-level, while the hierarchical structure of the text is neglected. Topic models, such as PLSA [13] and LDA [3], showed that words in the text generate various latent topics, and the perspectives of the latent topics demonstrate the semantic meaning of the text. Furthermore, topic models had shown great advantages in text understanding. In this paper, we will attempt to incorporate latent topics into RNN for text similarity modeling.

## 3 Fractional Latent Topic Based Recurrent Neural Network

In this section, we will introduce our fractional latent topic based RNN (FraLT-RNN) model for text similarity modeling in detail. For a better understanding,

we first give a brief introduction of the traditional RNN models as well as some notations used in this paper.

Given a pair of text as  $X = \{x_1, x_2, \dots, x_m\}$  and  $Y = \{y_1, y_2, \dots, y_n\}$ , we let  $\mathbf{x}_i$  and  $\mathbf{y}_j$  denote the embedding representations of the word  $x_i$  and  $y_j$  respectively. The traditional RNN approaches model each text separately. Take the text  $X$  as an example, we suppose the sequential hidden state as  $\mathbf{h}_1^x, \mathbf{h}_2^x, \dots, \mathbf{h}_m^x$  and each hidden state corresponds to a word in text  $X$ . Then, the hidden state can be calculated by:

$$\mathbf{h}_i^x = f(\mathbf{x}_i, \mathbf{h}_{i-1}^x), \quad (1)$$

where  $f$  can be defined with the long short-term memory (LSTM) model or the gated recurrent unit (GRU), and  $\mathbf{h}_i^x$  contains the context information from the first word to the current one [16]. After that, the text is represented by the attention [29] method over the hidden states as

$$\mathbf{H}_X = \sum_{i=1}^m \alpha_i \mathbf{h}_i^x, \quad (2)$$

where  $\alpha_i$  is the attention of  $\mathbf{h}_i^x$ . Finally, the similarity score is computed according to the two text representations ( $\mathbf{H}_X$  and  $\mathbf{H}_Y$ ) and a similarity function, such as cosine similarity.

### 3.1 Framework of FraLT-RNN

Previous work on RNN based text similarity modeling mainly learned the text representation in word-level, namely using the word representation for sentence representation and similarity modeling, while the hierarchical structures of a text pair are neglected. Topics of a text are generated by groups of words, and the distribution of the topics has been used for text similarity modeling. Referring to the topic models, it can be found that the semantic meaning of a text is more susceptible to the perspective of the topics it involves compared with the word-level features. In this paper, we facilitate the hierarchical structures of a text in similarity modeling and proposed a fractional latent topic based RNN (FraLT-RNN) model, which automatically learn latent topic and the topic attentive vector for text representation and similarity modeling.

The framework of our proposed FraLT-RNN model is shown in Fig. 1. Compared with the traditional RNN based models which learn the representation of a text with separated and independent words, our model takes a group of words in the text as a whole to generate latent topic which can better capture the hierarchical information and the topic-level interactions in the text pair. Specifically, we first generate the fractional latent topics for text  $\mathbf{Y}$  by mean of a fractional latent topic generator which involves the hidden states learned by RNN and a fractional calculus. With this step, the latent topics in the text will be captured and encoded. Then, a topic-wise attention gating mechanism is presented and embedded into our model, which controls the information flow between the text pair. As shown in Fig. 1, besides the topic representation, the word-level attention based text representations ( $\mathbf{H}_X$  and  $\mathbf{H}_Y$ ) are also the inputs of the gating to

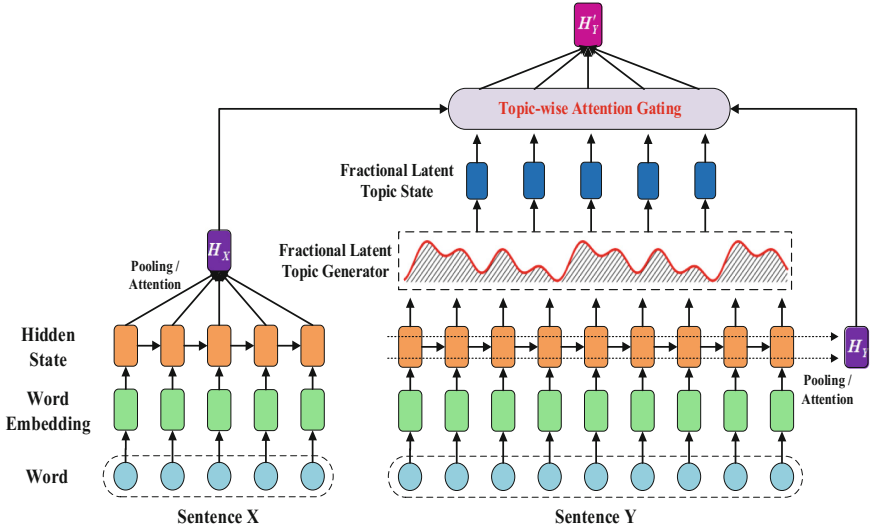


Fig. 1. A framework of fractional latent topic based RNN

help determine the perspectives of the topics. We will give a detailed description of each step in the following sections.

### 3.2 Fractional Latent Topic

Traditional topic models, such as PLSA [13] and LDA [3], usually use probabilities to analyze the latent topic distribution of the text, and have shown great advantages on text understanding and decoding. However, those models take the text as a bag of words and assume that words are independent with other, while the contextual information is neglected during latent topic generation. On the other hand, RNN models, such as LSTM and GRU, are excellent in processing and generating word sequence via taking the term dependency into consideration. Therefore, directly heap up the RNN model and traditional topic model for latent topic generation will miss contextual information such as sequential structure and term dependency in the text pair. To cope with this problem, we propose a fractional latent topic generator, which learns the latent topic of a text by means of the fractional calculus. Furthermore, the latent topic learned by the fractional latent topic generator is defined as the **fractional latent topic**. In the rest of this subsection, we will provide an insight into the fractional calculus, and then introduce the approach of the fractional latent topic generation.

**Fractional Calculus.** Fractional calculus, including fractional integral and fractional differential, which has been successfully used in image generation and denoising [2, 22]. Different from the ordinary integral and differential which conduct computing in integer order, the fractional calculus refines the computational

step into fractional order and considers the value of time-delayed states (namely the states before the current state) for the integral or differential implementation. Therefore, the fractional calculus has excellent characteristics of memory and heredity in processing sequential data. In this paper, we focus on the fractional integral, which is introduced in [17]. Suppose  $f(x)$  is a continuous function, then the Riemann-Liouville definition of the fractional integral in  $\alpha$  order is formulated as:

$$I^\alpha f(x) = \frac{1}{\Gamma(\alpha)} \int_0^x (x-t)^{\alpha-1} f(t) dt. \quad (3)$$

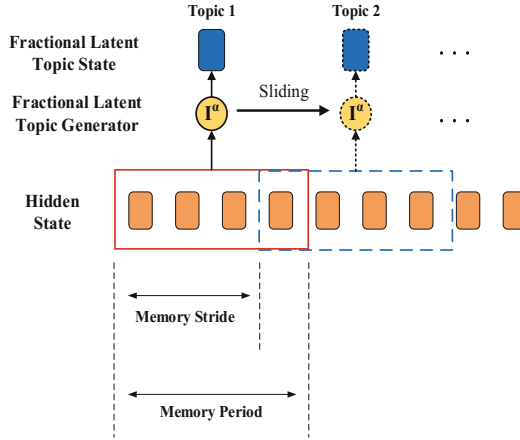
where  $I^\alpha$  stands for the fractional integral operator, and  $\Gamma(\cdot)$  is the gamma function which is defined by the Formula 4.

$$\Gamma(\alpha) = \int_0^{+\infty} t^{\alpha-1} e^{-t} dt \quad (4)$$

Taking the term  $\int_0^x (x-t)^{\alpha-1} f(t) dt$  in Formula 3 into consideration, the fractional integral of  $f(x)$  in order  $\alpha$  can be seen as the convolution operation between  $f(x)$  and  $x^{\alpha-1}$ , which involves all states of function  $f(\cdot)$  in the time interval  $[0, x]$ . If we take the  $f(\cdot)$  as a function of memory, 0 is the beginning of the memory and  $x$  is the end of the memory, then with this step, the fractional integral incorporates all states of  $f(\cdot)$  in the memory period  $[0, x]$  based on their convolutional interactions, and generates an overall perspective of the memory  $f(\cdot)$ . This also explains why the fractional integral operator has the capacity to process sequential data and has good performance in memory.

**Fractional Latent Topic Generation.** Since a text is usually composed by a word sequence, the latent semantic information of a text is closely related to the words semantic meanings and the contextual information such as term dependency and sequential structure [1, 14]. In this paper, therefore, we assume that the latent topic of a text can be reasoned and inferred from the occurred words' semantic representation and contextual information. It is notable that word semantic representation can be easily learned by the hidden state of a RNN model. Regarding to the contextual information, we first adopt a contextual window for text snippet sampling by sliding over the input word sequence, which has been reported to be effective in contextual feature extraction [27]. Then, we use a fractional latent topic generator derived from fractional calculus to aggregate contextual features and generate latent topics based on the text snippets. Figure 2 shows the procedure of the fractional latent topic generation.

*Text Snippet Sampling.* In RNN, hidden states are used to learn words representations. Since a text can be seen as a word sequence, the corresponding hidden state sequence can be regarded as a mirroring or projection of the given text. Therefore, we adopt a contextual window which slides from the beginning to the end of the hidden state sequence for text snippet sampling. As is shown in Fig. 2, the red box stands for the contextual window, the length of the contextual



**Fig. 2.** Procedure of the Fractional Latent Topic Generation. Where the red box stands for the contextual window, and the blue box indicates the location where the contextual window will sliding to in the next time. (Color figure online)

window is defined as the memory period and denoted by  $p$ , and the step length that the contextual window moving forward is defined as the memory stride. After the contextual window sliding over the whole text with a constant stride, we obtain all text snippets for the latent topic generation.

*Fractional Latent Topic Generator.* In the text snippet sampling step, we obtain various text snippets. The fractional latent topic generator, which is derived by fractional calculus, aims to learn a latent topic for each text snippet. It is notable that the fractional calculus is defined on a continuous function, while the hidden state in RNN is a discrete variable. Therefore, we are required to transform the fractional calculus to its discrete format. Formally, suppose a text snippet contains the hidden states as  $\mathbf{h}_{i-p+1}^y, \dots, \mathbf{h}_{i-1}^y, \mathbf{h}_i^y$ , then the hidden topic  $t_i$  in this memory period is calculated by the fractional latent topic generator as

$$t_i = \frac{1}{\Gamma(\alpha)} \sum_{j=i-p+1}^i (i+1-j)^{\alpha-1} \mathbf{h}_j^y, \quad (5)$$

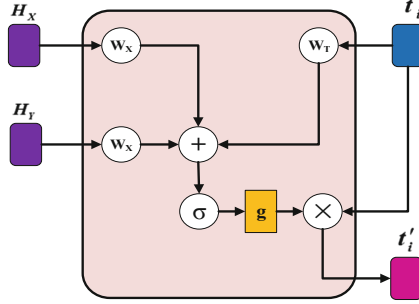
where  $p$  is the length of the memory period, and the fractional calculus order  $\alpha \in (0, 1]$  restricts the weights of hidden states. Different text snippets will generate different fractional latent topics, for example “Topic 1” and “Topic 2” in Fig. 2. With this step, we can obtain all fractional latent topics of the given text on the sampled text snippets.

### 3.3 Topic-Wise Attention

Intuitively, different topics tend to have different perspectives. Noting that the similarity of a text pair can be measured by the perspectives relevance of the



topics, we are motivated to decrease the perspective distance between topics in a relevant text pair, and increase the distance for an irrelevant pair, after generating the fractional latent topics for each text. In particular, we present a topic-wise attention gating mechanism for our model, which automatically rewards the perspectives of the topics with an attentive vector.



**Fig. 3.** Topic-wise attention gating mechanism

The structure of our topic-wise attention gating are shown in Fig. 3. Different from the traditional attention based RNN models, our gating mechanism makes full use of the information in two texts rather than a single one for the topic perspective rewarding. Specifically, to reward the perspective of a topic, our topic-wise attention mechanism mainly performs the following two steps, namely relevance measurement and perspective rewarding.

**Relevance Measurement.** The relevance measurement step measures the relevance between the fractional latent topic  $\mathbf{t}_i$  of text  $\mathbf{Y}$  and the text  $\mathbf{X}$ , which serves as a good criteria to decide how much information in the fractional latent topic should be rewarded. In particular, the whole semantic meaning of the text  $\mathbf{Y}$  should also be taken into consideration during the relevance calculation, since the fractional latent topic  $\mathbf{t}_i$  is generated on a segment of text  $\mathbf{Y}$ , instead of the whole  $\mathbf{Y}$ . More concretely, the relevance is formulated as:

$$\mathbf{g} = \sigma(\mathbf{W}_X \mathbf{H}_X + \mathbf{W}_Y \mathbf{H}_Y + \mathbf{W}_t \mathbf{t}_i + \mathbf{b}), \quad (6)$$

where  $\mathbf{W}_X$ ,  $\mathbf{W}_Y$  and  $\mathbf{W}_t$  are weight matrices,  $\mathbf{b}$  is a bias vector, and  $\sigma(\cdot)$  is an element-wise sigmoid function. It is worth noting that the obtained  $\mathbf{g}$  is a vector, which reflects the relevance in each hidden dimension.

**Perspective Rewarding.** In topic perspective rewarding step, the original fractional latent topic  $\mathbf{t}_i$  is refined by the rewarding vector  $\mathbf{g}$  obtained by Formula 6. With this step, the perspective tendency of a topic is modified to be more distinguishable for better text similarity modeling. The rewarded fractional latent topic is formulated as:

$$\mathbf{t}'_i = \mathbf{g} \odot \mathbf{t}_i, \quad (7)$$

where  $\odot$  denotes the element-wise multiplication, and  $\mathbf{t}'_i$  is the new rewarded hidden topic.

With the above two steps, the topic perspective gap in relevant text pair will be narrowed down, while it is opposite for the irrelevant one. It will have a big influence on the whole text modeling with the text is represented by

$$\mathbf{H}'_Y = \sum_{i=1}^{\tau} \mathbf{t}'_i. \quad (8)$$

## 4 Empirical Study

### 4.1 Datasets and Evaluation Metrics

To evaluate the effectiveness of our proposed model, we conduct experiments on three well-known text similarity tasks, namely question answering, paraphrase identification, and textual entailment.

**Question Answering.** Given a question and a list of candidate answers, the question answering task is to rank the candidates according to their similarities with the question. Two widely used datasets, namely TREC-QA and WikiQA, are adopted in our experiments. TREC-QA was created by Wang et al. [33] based on the QA track (8–13) data of Text REtrieval Conference. WikiQA [38] is an open domain QA dataset in which all answers were collected from the Wikipedia. Both TREC-QA and WikiQA have the train, development and test sets, and each sample is labeled as 1 or 0 to indicate whether the candidate answer is right or wrong for a given question. The statistics of the datasets are presented in Table 1. The performance of answer selection is usually measured by the mean average precision (MAP) and mean reciprocal rank (MRR) [25].

**Paraphrase Identification.** The paraphrase identification task can be treated as a binary classification problem, and the goal is to judge whether two texts are paraphrases or not according to their similarity. We utilize the Microsoft Research Paraphrase corpus (MSRP) [5] for experiment, which is constructed from a large corpus of temporally and topically clustered news articles. The MSRP dataset contains 4,076 sentence pairs in the training set, and 1,725 ones in the test set. Each text pair is labeled with 1 or 0 to indicate whether the two text are paraphrases or not. To evaluate the performance, two widely used metrics, namely accuracy (Acc) and F1 score are adopted [39].

**Textual Entailment.** For a sentence pair, one of a sentence can be seen as the premise and the other as the hypothesis. The textual entailment task is to judge whether the hypothesis can be inferred by the premise according to their similarity. We use the Multi-Genre Natural Language Inference (MultiNLI) corpus for experiment, which is a crowd-sourced collection of sentence pairs annotated

**Table 1.** Statistics of the Datasets. “Avg QL”, “Avg AL”, “Avg Para1L”, “Avg Para2L”, “Avg Sent1L” and “Avg Sent2L” denote the average length of questions, answers, the first paragraphs, the second paragraphs, the first sentences and the second sentences respectively.

Answer selection	Dataset		# of Questions	Avg QL	Avg AL
	TREC-QA	train	1162	7.57	23.21
		dev	65	8.00	24.9
		test	68	8.63	25.61
	WikiQA	train	873	7.16	25.29
		dev	126	7.23	24.59
est		243	7.26	24.59	
Paraphrase Identification	Dataset		# of Paragraph Pair	Avg Para1L	Avg Para2L
	MSRP	train	4077	18.99	18.93
		test	1725	18.82	18.80
Textual Entailment	Dataset		# of Sentence Pair	Avg Sent1L	Avg Sent2L
	MultiNLI	train	392,702	19.91	10.12
		matched	10,000	19.40	10.08
		mismatched	10,000	19.90	10.98

with textual entailment information. In MultiNLI, the relationship between a sentence pair is classified into three categories, namely neutral, contradiction, and entailment. During our experiments, we assign the value of  $-1$ ,  $0$ ,  $1$  to the label of neutral, contradiction, and entailment respectively. Furthermore, this corpus has served as the basis for the shared task of the RepEval 2017 Workshop<sup>1</sup> at EMNLP in Copenhagen. and the evaluation metric used in this corpus is accuracy (Acc) [36].

## 4.2 Training

We use the bidirectional LSTM (BLSTM) [9] model as the function in Formula 1 to obtain the original hidden states, which can effectively mitigate the gradient vanish problem. Then, we utilize the Manhattan distance similarity function with  $l1$  norm and restrict it to a range of  $[0, 1]$  for text similarity calculation [20]:

$$s(X, Y) = \exp(-\|\mathbf{H}'_X - \mathbf{H}'_Y\|_1) \quad (9)$$

where,  $\mathbf{H}'_X$  and  $\mathbf{H}'_Y$  are text representations learned by the proposed FraLT-RNN model. The predicted probability of a text pair labeled as 1 or 0 is defined according to the relevance score:  $\hat{p}(c = 1|X, Y) = s(X, Y)$  and  $\hat{p}(c = 0|X, Y) = 1 - s(X, Y)$ .

<sup>1</sup> <https://repeval2017.github.io/shared/>.

For each text pair, the loss function is defined by the cross-entropy of the predicted and true label distributions for training:

$$L(X, Y; c) = - \sum_{j=0}^{C-1} p(c = j|X, Y) \log \hat{p}(c = j|X, Y) \quad (10)$$

where  $C$  is the number of classes, and  $p(c = 1|X, Y)$  is the gold probability of label  $c$ , which equals to 1 with ground truth and otherwise is 0.

### 4.3 Parameter Settings

We implement the proposed FraLT-RNN model by using TensorFlow. The optimization is relatively straightforward with standard back-propagation [24]. We apply stochastic gradient descent method Adagrad [6] with mini-batches (64 in size), which can be easily parallelized on single machine with multi-cores. The rectifier linear unit  $ReLU = \max(0, x)$  is adopted as the activation function, which is a common choice in the deep learning literature [19]. For regularization, we use dropout [11] strategy for our model, and the dropout rate is selected from [0.0, 0.1, 0.2, 0.5]. Regarding to the word embeddings, we adopt the pre-trained 100-dimensional GloVe word vectors<sup>2</sup>, which are trained based on the global word co-occurrence [21]. Moreover, the fractional integral order  $\alpha$  is valued from 0.1 to 1 with the stride of 0.1, and the memory period is selected in the set [2, 3, 4, 5, 6, 7, 8, 9, 10].

## 5 Experimental Results and Analyses

### 5.1 Effectiveness of FraLT-RNN

To investigate the effect of our FraLT-RNN model, the BLSTM based RNN model which does not involve any topic information, and the recently proposed word-level attention mechanism [29] are utilized for comparisons. Table 2 shows the performance of various models for question answering, paraphrase identification, and textual entailment tasks. It is observed that we achieve significant improvements over classical BLSTM and attention based BLSTM models on all datasets, by incorporating fractional latent topic into text representation. It is also notable that the classical BLSTM model relies more on the attention mechanism to capture the salient information for text similarity modeling. However, the attention method mainly focuses on measuring the weight of each hidden state, while does not pay specific attention to the surrounding context of the words in a text pair. Moreover, the attentive weight is produced after obtaining all the hidden states, which neglects the internal interactions and hierarchical structure of the text during hidden state generation. In contrast, our proposed FraLT-RNN model can explicitly capture the internal relations and the hierarchical features between two texts, by incorporating the fractional latent topics

<sup>2</sup> <http://nlp.stanford.edu/data/glove.6B.zip>.

and topic-wise attentions for text representation. Therefore, our FraLT-RNN model integrated can yield better performance than the traditional attention mechanism.

**Table 2.** Comparison with Various RNN Models. “BLSTM” stands for BLSTM with no more optimization, and “A-BLSTM” stands for traditional word attention based BLSTM. “\*” and “+” imply significant improvements over “BLSTM” and “A-BLSTM” respectively.

Model	TREC-QA		WikiQA		MSRP		MultiNLI	
	MAP	MRR	MAP	MRR	Acc(%)	F1	Matched (Acc %)	Mismatched (Acc %)
BLSTM	0.6487	0.6991	0.6581	0.6691	73.6	81.8	67.5	67.1
A-BLSTM	0.7369*	0.8208*	0.7258*	0.7394*	75.4*	82.7*	71.1*	70.8*
FraLT-RNN	<b>0.8359</b> **	<b>0.8962</b> **	<b>0.7401</b> **	<b>0.7519</b> **	<b>81.2</b> **	<b>87.5</b> **	<b>81.9</b> **	<b>81.3</b> **

## 5.2 Comparison with Recent Progress

In addition to the classical BLSTM model, we compare our model with the recent progress in question answering, paraphrase identification and textual entailment.

**Table 3.** Performance comparisons on TREC-QA

System	MAP	MRR
Wang, Liu, and Zhao 2016 [31]	0.7369	0.8208
Wang and Ittcheriah 2015 [34]	0.7460	0.8200
Santos et al. 2016 [25]	0.7530	0.8511
Wang, Mi, and Ittycheriah 2016 [35]	0.7714	0.8447
Chen et al. 2018 [4]	0.8227	0.8886
FraLT-RNN	<b>0.8359</b>	<b>0.8962</b>

**Results on Question Answering.** Table 3 and Table 4 summarize the results on TREC-QA and WikiQA respectively. [25, 31, 40] are the recent attention based models that focus on the word-level attentive text representations. It is observed that our proposed model achieves the new state-of-the-art performance on both TREC-QA and WikiQA. Specifically, we outperform the best results on TREC-QA and WikiQA with absolute improvements of 0.132 and 0.0043 in terms of MAP, and 0.0076 and 0.0069 in terms of MRR. Regarding to the word alignment models [4, 34, 35], which take the aligned words and the neighboring texts into consideration for text representation, our model is also much more effective and does not rely on the laboursome feature engineering. This is mainly owing to the hierarchical structure embedded in our model, namely the latent topics, which are neglected in the above models.

**Table 4.** Performance comparisons on WikiQA

System	MAP	MRR
Santos et al. 2016 [25]	0.6886	0.6957
Yin et al. 2015 [40]	0.6921	0.7108
Wang, Mi, and Ittycheriah 2016 [35]	0.7058	0.7226
Wang, Liu and Zhao 2016 [31]	0.7341	0.7418
Chen et al. 2018 [4]	0.7358	0.7450
FraLT-RNN	<b>0.7401</b>	<b>0.7519</b>

**Table 5.** Performance comparisons on MSRP

System	Acc	F1
Hu et al. 2014 [15]	69.9	80.9
Socher et al. 2011 [28]	76.8	83.6
Yin and Schutze 2015 [39]	78.1	84.4
Chen et al. 2018 [4]	77.3	84.0
He, Gimpel, and Lin 2015 [10]	78.6	84.7
FraLT-RNN	<b>81.2</b>	<b>87.5</b>

**Results on Paraphrase Identification.** The results from recent work on MSRP are summarized in Table 5. [39] presented a convolutional neural network based deep learning architecture, which modeled interaction features at multiple levels of granularity. However, their model relied much on the pretraining step. In [10], a similar model was proposed, which also used a CNN model for feature extraction at a multiplicity of perspectives. We observe that our model also achieve the best results among the existing work. Furthermore, our model automatically generates fractional latent topics with a fractional latent topic generator, which requires no more parameter besides the fractional order  $\alpha$  and has a lower computation complexity.

**Table 6.** Performance comparisons on MultiNLI

System	Matched	Mismatched
Williams, Nangia, and Bowman 2017 [37]	72.3	72.1
Gong, Luo, and Zhang 2018 [8]	78.8	77.8
Tay, Tuan, and Hui 2017 [30]	78.7	77.9
Kim, Kang, and Kwak 2018 [18]	79.1	78.4
Radford et al. 2018 [23]	<b>82.1</b>	<b>81.4</b>
FraLT-RNN	81.9	81.3

**Results on Textual Entailment.** Table 6 shows the comparison with recent work on matched and mismatched problems of MultiNLI collection. It can be seen that our model outperforms most of the recent work, and can be comparable to if no better than the state-of-the-art model [23]. [23] makes use of task-aware input transformations to achieve effective transfer for natural language understanding. However, their model requires to learn a pre-trained model at first, and then tunes parameters in the pre-trained model for a better performance. During the model tuning step, numerous parameters need to be learned. In contrast,

our FraLT-RNN model does not need a pre-trained model and only requires one parameter, namely the fractional integral order, besides the RNN parameters during model learning.

### 5.3 Influence of the Parameters in Fractional Latent Topic Generation

For the proposed FraLT-RNN model, there are two components in fractional latent topic generation, namely the text snippet sampling and the fractional latent topic generator. The memory period and the fractional calculus order is the main parameter in the text snippet sampling and the fractional latent topic generator respectively. We conduct experiments on the four datasets to investigate the influence of the two parameters.

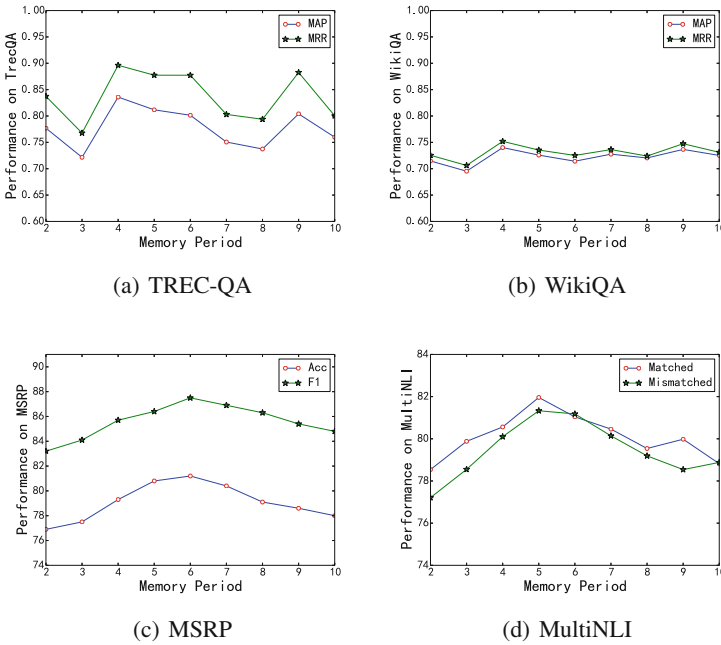


Fig. 4. Influence of the memory period.

Figure 4 show the influence of the memory period on the FraLT-RNN. The memory period decides the range of the term dependency, i.e. a large memory period implies a long-term dependency. It can be seen that TREC-QA is more sensitive on the memory period compared with the other three datasets. To ensure a better and steady performance of FraLT-RNN, it is recommended to select the memory period value of the contextual window in the set [4, 5, 6, 7]. Regarding to the fractional calculus order  $\alpha$ , Fig. 5 illustrates its influence on

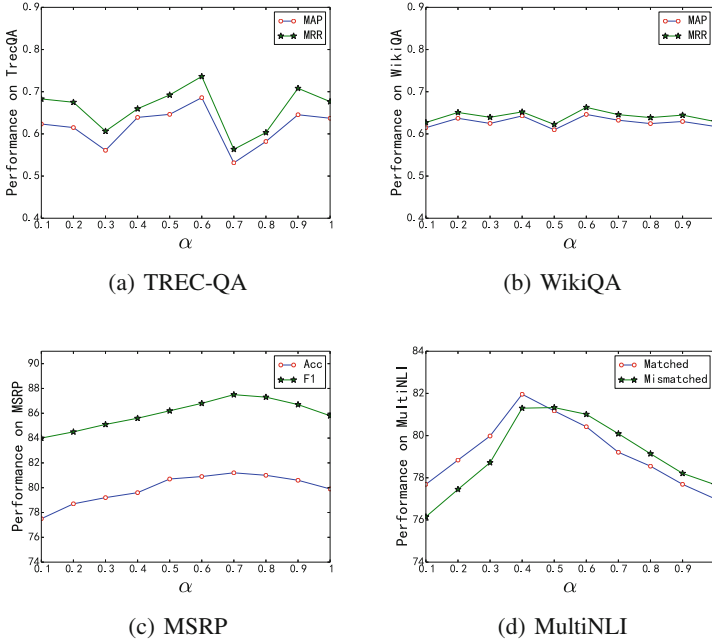


Fig. 5. Influence of the fractional calculus order  $\alpha$ .

the proposed FraLT-RNN model. It is interesting to find that TREC-QA is also more sensitive on the fractional calculus order compared with the other three datasets, which is consistent with the memory period parameter. Furthermore, to obtain a more reliable result, it is recommended to assign the value of fractional calculus order  $\alpha$  in the interval  $[0.4, 0.8]$ .

## 6 Conclusion and Future Work

In this paper, we proposed a fractional latent topic based RNN model, which incorporates the hierarchical structures of the text during text representation. In particular, we provide a novel latent topic generation approach, which is implemented by means of the fractional calculus. To the best of our knowledge, this is the first attempt to apply the fractional calculus in natural language processing. Experiments on four benchmark datasets, namely TREC-QA and WikiQA for question answering, MSRP for paraphrase identification, and MultiNLI for textual entailment show the great advantages of our proposed model. It is notable that we achieve the new state-of-the-art results on TREC-QA, WikiQA, and MSRP. It is also interesting to find that the TREC-QA is more sensitive to the parameters of fractional latent topic generator. In the future, we will investigate how to apply the fractional calculus into more natural language processing tasks and deep learning models to cope with the time series and hierarchical structure problems.



**Acknowledgement.** This research is funded by the Science and Technology Commission of Shanghai Municipality (No. 18511105502), Shanghai Municipal Commission of Economy and Informatization (No. 170513) and Xiaoi Research. The computation is performed in the Supercomputer Center of ECNU. The second author is the corresponding author.

## References

1. Al-Anzi, F.S., AbuZeina, D.: Toward an enhanced Arabic text classification using cosine similarity and latent semantic indexing. *J. King Saud University-Comput. Inf. Sci.* **29**(2), 189–195 (2017)
2. Bai, J., Feng, X.C.: Fractional-order anisotropic diffusion for image denoising. *IEEE Trans. Image Process.* **16**(10), 2492–2502 (2007)
3. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. *J. Mach. Learn. Res.* **3**(Jan), 993–1022 (2003)
4. Chen, Q., Hu, Q., Huang, J.X., He, L.: CA-RNN: using context-aligned recurrent neural networks for modeling sentence similarity. In: *AAAI* (2018)
5. Dolan, B., Quirk, C., Brockett, C.: Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources. In: *Proceedings of the 20th International Conference on Computational Linguistics*, p. 350. Association for Computational Linguistics (2004)
6. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* **12**(Jul), 2121–2159 (2011)
7. Fang, H., Wu, F., Zhao, Z., Duan, X., Zhuang, Y., Ester, M.: Community-based question answering via heterogeneous social network learning. In: *Thirtieth AAAI Conference on Artificial Intelligence* (2016)
8. Gong, Y., Luo, H., Zhang, J.: Natural language inference over interaction space. *arXiv preprint [arXiv:1709.04348](https://arxiv.org/abs/1709.04348)* (2017)
9. Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Netw.* **18**(5–6), 602–610 (2005)
10. He, H., Gimpel, K., Lin, J.: Multi-perspective sentence similarity modeling with convolutional neural networks. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1576–1586 (2015)
11. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint [arXiv:1207.0580](https://arxiv.org/abs/1207.0580)* (2012)
12. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.* **9**(8), 1735–1780 (1997)
13. Hofmann, T.: Unsupervised learning by probabilistic latent semantic analysis. *Mach. Learn.* **42**(1–2), 177–196 (2001)
14. Hofmann, T.: Probabilistic latent semantic indexing. In: *ACM SIGIR Forum*, vol. 51, pp. 211–218. ACM (2017)
15. Hu, B., Lu, Z., Li, H., Chen, Q.: Convolutional neural network architectures for matching natural language sentences. In: *Advances in Neural Information Processing Systems*, pp. 2042–2050 (2014)
16. Jozefowicz, R., Zaremba, W., Sutskever, I.: An empirical exploration of recurrent network architectures. In: *International Conference on Machine Learning*, pp. 2342–2350 (2015)

17. Kilbas, A.A.A., Srivastava, H.M., Trujillo, J.J.: Theory and Applications of Fractional Differential Equations, vol. 204. Elsevier Science Limited, Amsterdam (2006)
18. Kim, S., Hong, J.H., Kang, I., Kwak, N.: Semantic sentence matching with densely-connected recurrent and co-attentive information. arXiv preprint [arXiv:1805.11360](https://arxiv.org/abs/1805.11360) (2018)
19. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. *Nature* **521**(7553), 436–444 (2015)
20. Mueller, J., Thyagarajan, A.: Siamese recurrent architectures for learning sentence similarity. In: *AAAI*, vol. 16, pp. 2786–2792 (2016)
21. Pennington, J., Socher, R., Manning, C.: GloVe: global vectors for word representation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543 (2014)
22. Pu, Y., Wang, W., Zhou, J., Wang, Y., Jia, H.: Fractional differential approach to detecting textural features of digital image and its fractional differential filter implementation. *Sci. China Series F Inf. Sci.* **51**(9), 1319–1339 (2008)
23. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training. [https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language\\_understanding\\_paper.pdf](https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf) (2018)
24. Rumelhart, D.E., Hinton, G.E., Williams, R.J., et al.: Learning representations by back-propagating errors. *Cogn. Mod.* **5**(3), 1 (1988)
25. dos Santos, C.N., Tan, M., Xiang, B., Zhou, B.: Attentive pooling networks. *CoRR*, abs/1602.03609 **2**(3), 4 (2016)
26. Severyn, A., Moschitti, A.: Learning to rank short text pairs with convolutional deep neural networks. In: *SIGIR*, pp. 373–382 (2015)
27. Shen, Y., He, X., Gao, J., Deng, L., Mesnil, G.: A latent semantic model with convolutional-pooling structure for information retrieval. In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pp. 101–110. ACM (2014)
28. Socher, R., Huang, E.H., Pennin, J., Manning, C.D., Ng, A.Y.: Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In: *Advances in Neural Information Processing Systems*, pp. 801–809 (2011)
29. Tan, M., Santos, C.D., Xiang, B., Zhou, B.: LSTM-based deep learning models for non-factoid answer selection. arXiv preprint [arXiv:1511.04108](https://arxiv.org/abs/1511.04108) (2015)
30. Tay, Y., Tuan, L.A., Hui, S.C.: A compare-propagate architecture with alignment factorization for natural language inference. arXiv preprint [arXiv:1801.00102](https://arxiv.org/abs/1801.00102) (2017)
31. Wang, B., Liu, K., Zhao, J.: Inner attention based recurrent neural networks for answer selection. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, pp. 1288–1297 (2016)
32. Wang, D., Nyberg, E.: A long short-term memory model for answer sentence selection in question answering. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, vol. 2, pp. 707–712 (2015)
33. Wang, M., Smith, N.A., Mitamura, T.: What is the jeopardy model? A quasi-synchronous grammar for QA. In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)* (2007)
34. Wang, Z., Ittycheriah, A.: FAQ-based question answering via word alignment. arXiv preprint [arXiv:1507.02628](https://arxiv.org/abs/1507.02628) (2015)

35. Wang, Z., Mi, H., Ittycheriah, A.: Sentence similarity learning by lexical decomposition and composition. arXiv preprint [arXiv:1602.07019](https://arxiv.org/abs/1602.07019) (2016)
36. Williams, A., Nangia, N., Bowman, S.: A broad-coverage challenge corpus for sentence understanding through inference. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pp. 1112–1122. Association for Computational Linguistics (2018). <http://aclweb.org/anthology/N18-1101>
37. Williams, A., Nangia, N., Bowman, S.R.: A broad-coverage challenge corpus for sentence understanding through inference. arXiv preprint [arXiv:1704.05426](https://arxiv.org/abs/1704.05426) (2017)
38. Yang, Y., Yih, W.T., Meek, C.: WikiQA: a challenge dataset for open-domain question answering. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 2013–2018 (2015)
39. Yin, W., Schütze, H.: Convolutional neural network for paraphrase identification. In: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 901–911 (2015)
40. Yin, W., Schütze, H., Xiang, B., Zhou, B.: ABCNN: attention-based convolutional neural network for modeling sentence pairs. arXiv preprint [arXiv:1512.05193](https://arxiv.org/abs/1512.05193) (2015)
41. Zhang, X., Li, S., Sha, L., Wang, H.: Attentive interactive neural networks for answer selection in community question answering. In: AAAI, pp. 3525–3531 (2017)
42. Zhao, Z., Lu, H., Zheng, V.W., Cai, D., He, X., Zhuang, Y.: Community-based question answering via asymmetric multi-faceted ranking network learning. In: AAAI, pp. 3532–3539 (2017)



# Efficiently Mining Maximal Diverse Frequent Itemsets

Dingming Wu<sup>1</sup>(✉), Dexin Luo<sup>1</sup>, Christian S. Jensen<sup>2</sup>,  
and Joshua Zhexue Huang<sup>1</sup>

<sup>1</sup> College of Computer Science and Software Engineering, Shenzhen University,  
Shenzhen, China

{dingming,zx.huang}@szu.edu.cn, luodexin2016@email.szu.edu.cn

<sup>2</sup> Department of Computer Science, Aalborg University, Aalborg, Denmark  
csj@cs.aau.dk

**Abstract.** Given a database of transactions, where each transaction is a set of items, maximal frequent itemset mining aims to find all itemsets that are frequent, meaning that they consist of items that co-occur in transactions more often than a given threshold, and that are maximal, meaning that they are not contained in other frequent itemsets. Such itemsets are the most interesting ones in a meaningful sense. We study the problem of efficiently finding such itemsets with the added constraint that only the top- $k$  most diverse ones should be returned. An itemset is diverse if its items belong to many different categories according to a given hierarchy of item categories. We propose a solution that relies on a purposefully designed index structure called the FP\*-tree and an accompanying bound-based algorithm. An extensive experimental study offers insight into the performance of the solution, indicating that it is capable of outperforming an existing method by orders of magnitude and of scaling to large databases of transactions.

**Keywords:** Frequent itemsets · Diversification · Algorithm

## 1 Introduction

Frequent itemset mining [2] is important data analysis functionality. The prototypical application is supermarket basket analysis that allows a retailer to learn which items are commonly bought together. For instance, it might be found that “bread” and “milk” are often bought together. A major issue in frequent itemset mining is the consideration of a huge number of itemsets, many of which are eventually found to be insignificant. Hence, researchers have made efforts to mine different constraint-based frequent itemsets, considering different kind of itemsets, including closed [13], maximal [4], periodic [19], top- $k$  [14], cost (utility) [15], sequential [12], weighted [20], and diverse [17, 18] itemsets.

Diverse frequent itemset mining [17, 18] targets scenarios where it may be useful to give priority to frequent itemsets with items belonging to different

item categories. A measure called DiverseRank was introduced to quantify the extent to which items in a set belong to multiple categories. The existing algorithm [17] is inefficient for computing the diverse frequent itemsets on large data sets. The algorithm first extracts all frequent itemsets using the state-of-the-art algorithm [9] and then extracts the possibly very small subset of diverse itemsets from the frequent itemsets. It is a waste of time computing frequent itemsets that are later eliminated because they are not diverse.

We combine the maximality and diversity constraints and study the problem of efficiently finding the top- $k$  maximal diverse frequent itemsets (MDFIs). Compared to just finding diverse frequent itemsets, the maximality constraint is able to reduce the number of discovered itemsets, since a frequent itemset is maximal if none of its supersets are frequent. To support MDFI mining efficiently on large data sets, we propose the FP\*-tree, a variant of the FP-tree [9] that not only is able to store compactly the necessary information for MFI computation, but also contains a posting list for each item, which makes it possible to construct supersets for maximal frequent itemsets (MFIs). We show that these supersets can cover all the MFIs in the data set. However, the function for computing the diversity score is non-monotonous. Therefore, the diversity score of those supersets cannot be used as upper bounds on the diversity scores of the covered MFIs. Hence, we propose an algorithm that derives upper bounds on the diversity scores of the MFIs to be computed. Using the FP\*-tree, we present a bound-based algorithm that is able to return the top- $k$  MDFIs while computing only some of the MFIs in the data set. Unlike existing methods that mine the MFIs in descending order of item frequency, the proposed algorithm adopts a new ordering based on the upper bounds on the diversity score for the MDFI computation, so that the top- $k$  result can be obtained by computing only a few candidate MFIs. The proposed algorithm is compared to a basic algorithm that extends two existing algorithms. The performance evaluation on a real data set shows that the proposed algorithm on the FP\*-tree outperforms the basic algorithm by up to several orders of magnitude.

The rest of the paper is organized as follows. First, Sect. 2 presents preliminaries and defines the paper’s problem formally. Then, Sect. 3 presents the FP\*-tree and the accompanying bound-based algorithm. Experimental results are reported in Sect. 4, and Sect. 5 reviews related work. Finally, Sect. 6 concludes and offers research directions.

## 2 Preliminaries and Problem Definition

Let  $I$  be a finite set of items,  $I = \{i_1, i_2, \dots, i_m\}$ . A database  $\mathcal{D} = \{T_1, T_2, \dots, T_n\}$  is a set of transactions, where each transaction  $T_j \in \mathcal{D}$  ( $1 \leq j \leq n$ ) is a subset of  $I$  and is assigned a unique identifier  $j$ . An itemset  $X = \{i_1, i_2, \dots, i_l\}$  is a set of  $l$  items, where  $i_j \in I$  ( $1 \leq j \leq l$ ) and  $l$  is the length of  $X$ . An itemset  $X$  is contained in a transaction  $T$  if  $X \subseteq T$ . The **support**<sup>1</sup>  $s(X)$  of an itemset  $X$  is the

<sup>1</sup> The support of an itemset can be also defined as the fraction of transactions that contain it. For simplicity, we use the count of transactions, which is equivalent when database  $\mathcal{D}$  is fixed.

number of transactions containing  $X$  in  $\mathcal{D}$ . Given  $1 \leq \sigma \leq |\mathcal{D}|$ , an itemset  $X$  is called  $\sigma$ -**frequent** in  $\mathcal{D}$  if  $s(X) \geq \sigma$ . A  $\sigma$ -frequent itemset  $X$  in  $\mathcal{D}$  is a **maximal  $\sigma$ -frequent itemset** (MFI) in  $\mathcal{D}$  if no  $\sigma$ -frequent itemset  $X'$  exists in  $\mathcal{D}$  such that  $X' \supset X$  [4].

A category tree  $CT$  is a tree structure, where non-leaf nodes correspond to categories and leaf nodes are items in  $I$ . Each internal node is the sub-category of its parent node. Each item (leaf node)  $i_j \in I$  ( $1 \leq j \leq m$ ) belongs to the category of its parent node. Nodes close to the root correspond to general categories, while nodes close to leaf nodes correspond to specialized categories. The height  $h$  of a category tree is the length of the longest path from the root to a leaf node. The height of the root is  $h$ , and the height of a leaf node is 0. The level of a node is the length of the path from the node to the root. The level of the root is then 0, and the level of a leaf node is  $h$ . We consider only balanced category trees, i.e., paths from the root to a leaf node have the same length. Including a category tree in maximal frequent itemset mining makes it possible to distinguish between itemsets with similar items and itemsets with dissimilar items.

**Definition 1.** Let  $X$  be an itemset, let  $l$  be a level in the category tree, where  $0 \leq l \leq h$ . We define  $GP(X, l)$  to be the **generalized pattern** [17] of  $X$  at level  $l$  as follows.  $GP(X, h) = X$ , and  $GP(X, l), l < h$ , is obtained by replacing each item in  $GP(X, l+1)$  with its corresponding parent at level  $l$  with duplicates removed, if any.

**Definition 2.** The **Merging Factor** [17]  $MF(X, l)$  of itemset  $X$  at a level  $l$  depends on the number of items getting merged when the pattern is moved from the immediate lower level ( $l+1$ ) to level  $l$  in the category tree

$$MF(X, l) = \frac{|GP(X, l)| - 1}{|GP(X, l+1)| - 1}, \quad 0 \leq l \leq h-1 \quad (1)$$

**Definition 3.** The **Proportional Level Factor** [17]  $PLF(l)$  of level  $l$  is defined as:

$$PLF(l) = \frac{2(h-l)}{(h-1)h}, \quad 1 \leq l \leq h-1, \quad h > 1 \quad (2)$$

**Definition 4.** The **diversity score**  $div(X)$  of itemset  $X$  is defined as the *DiverseRank* [17] of  $X$ .

$$div(X) = \sum_{l=h-1}^{s+1} PLF(l)MF(X, l), \quad (3)$$

where  $s$  is the level at which  $|GP(X, s)| = 1$ .

A generalized pattern GP of an itemset represents the itemset in a category space. The smaller the level of a category is, the more general the category is. The merging factor reflects how fast the size of the GP is reduced when moving upward in the category tree. The *PLF* assigns weights to levels. The contributions of the levels near the root should be larger than those of the levels

near the leaf nodes. The diversity score of a frequent itemset ranges from  $[0, 1]$ . If all the items in a frequent pattern have the same immediate parent, the score is 0. On the other hand, if all the items in a frequent itemset have only the root as the common ancestor, the score is 1. The higher the score is, the more diverse the itemset is.

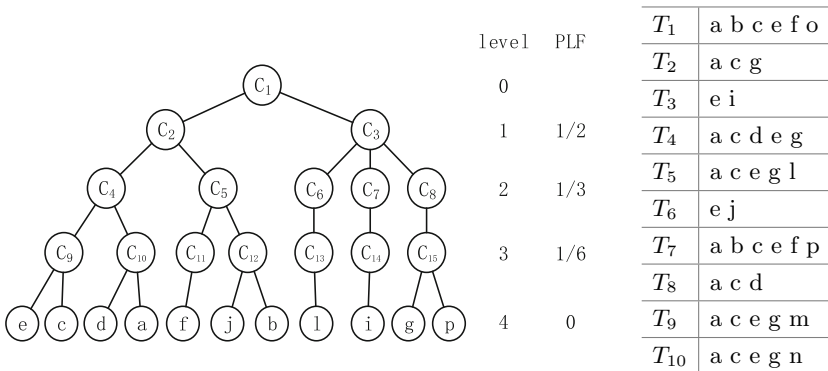
**Example 1.** Figure 1 shows an example of a category tree with height  $h = 4$ . Consider itemset  $X = \{c, e, f\}$ . The generalized pattern of  $X$  at level 3 is  $GP(X, 3) = \{C_9, C_{11}\}$ . Items  $c$  and  $e$  both have  $C_9$  as parent at level 3 and because the parent of item  $f$  at level 3 is  $C_{11}$ . The merging factor  $MF(X, 3)$  of itemset  $X$  at level 3 is  $(|GP(X, 3)| - 1) / (|GP(X, 4)| - 1) = (2 - 1) / (3 - 1) = 0.5$ . The proportional level factor (PLF) at each level is shown in Fig. 1. The diversity score of itemset  $X$  is  $div(X) = PLF(3)MF(X, 3) + PLF(2)MF(X, 2) = (1/6) \cdot 0.5 + (1/3) \cdot 1 = 0.42$  because the generalized pattern of  $X$  at level 1 is  $\{C_2\}$  and  $|GP(X, 1)| = 1$ , meaning that  $s = 1$ .

**Definition 5.** An itemset  $X$  is called a **top- $k$  maximal diversified  $\sigma$ -frequent itemset** ( $k$ MDFI) in  $\mathcal{D}$  if it satisfies two conditions:

1.  $X$  is a maximal  $\sigma$ -frequent itemset in  $\mathcal{D}$ .
2. There are fewer than  $k$  maximal  $\sigma$ -frequent itemsets in  $\mathcal{D}$  with diversity scores that exceed  $div(X)$ .

**Problem Statement.** Given a database  $\mathcal{D}$  of transactions, a category tree  $CT$ , a user-defined support threshold  $\sigma$ , and a desired number of itemsets  $k$ , the problem is to find efficiently a set of **top- $k$  maximal diversified  $\sigma$ -frequent itemsets** ( $k$ MDFIs) in  $\mathcal{D}$ , i.e., to discover efficiently  $k$  maximal  $\sigma$ -frequent itemsets with the highest diversity scores in  $\mathcal{D}$ .

**Example 2.** Consider the transactional database  $\mathcal{D}$  in Table 1 and the category tree  $CT$  in Fig. 1. Let  $\sigma = 2$ . The top-2 maximal diversified 2-frequent itemsets are  $X_1 = \{a, c, e, g\}$  and  $X_2 = \{a, b, c, e, f\}$  with diversity scores  $div(X_1) = 0.78$  and  $div(X_2) = 0.24$ .



**Fig. 1.** Example category tree

**Table 1.** Transactions

### 3 Bound-Based Mining Algorithm

We present a method that is able to efficiently compute the  $k$ MDFIs in large databases. Section 3.1 introduces the FP\*-tree that stores information necessary to enable  $k$ MDFI mining. Section 3.2 presents the bound-based algorithm that uses the FP\*-tree for mining  $k$ MDFIs. Section 3.3 derives bounds on the diversity scores of MDFIs.

#### 3.1 The FP\*-Tree

The FP-tree [9] is an index on transactions that enables frequent itemset mining. It consists of a tree structure and a header table. Each row in the header table stores a frequent item, its frequency, and a pointer to a tree node. The rows are sorted in non-increasing order of their frequencies. Ties are broken arbitrarily if multiple items have the same frequency. The header table for the transactions in Table 1 is shown in Fig. 2, given  $\sigma = 2$ . The tree structure stores all information necessary for mining frequent itemsets in a compact manner. It is built in the following way. Initially, the tree contains only one root node. Tree nodes are created and updated as transactions are scanned one by one. A branch in the FP-tree stores items that appear in the same transactions, and the nodes along a branch occur in the same order of the corresponding items in the header table. Overlapping itemsets are represented by the sharing of prefixes of the corresponding branches. For each transaction, the items included are sorted using the item order in the header table. Consider the transactions in Table 1. The scan of the first transaction leads to the construction of the first branch of the tree:  $(e : 1), (c : 1), (a : 1), (b : 1), (f : 1)$ . Item  $o$  is removed, since it is not contained in the header table, meaning that  $o$  is infrequent. For the fourth transaction, since its (ordered) frequent item list  $e, c, a, g, d$  shares a common prefix  $e, c, a$  with the existing path, the count of each node along the prefix is incremented by 1, and a new node  $(g : 1)$  is created and linked as a child of  $(a : 2)$  and another new node  $(d : 1)$  is created and linked as the child of  $(g : 1)$ . The tree-structure on the right side of Fig. 2 is the FP-tree built on the transactions in Table 1.

Before presenting the FP\*-tree, we define the rank  $r(i)$  of item  $i$  in the header table as the position of  $i$  in the table. The rank of the first item is 1, and if item  $i$  occurs before item  $i'$ ,  $r(i) < r(i')$ . Let  $T(i)$  be the set of transactions that contain  $i$ . The FP\*-tree extends the FP-tree [9] by adding a posting list  $L(i)$  of (item, counter) pairs for each item  $i$  in the header table. A pair  $(i', counter)$  for item  $i$  indicates that  $i$  and  $i'$  co-occur  $counter$  times in transactions. An item  $i'$  must satisfy two conditions to be included in the posting list of item  $i$ :  $r(i') < r(i)$  and  $T(i) \cap T(i') \neq \emptyset$ .

**Example 3.** Figure 2 shows the FP\*-tree of the transactions in Table 1 for  $\sigma = 2$ . The header table and the tree structure are the same as in the FP-tree. The posting list of item  $e$  is empty, since it is the first item in the header table and no item has lower rank. The posting list of item  $a$  consists of pairs  $(c, 8)$  and  $(e, 6)$ , meaning that (1)  $c$  and  $e$  have lower rank than  $a$ , (2) items  $c$  and  $a$  co-occur 8 times in transactions, and (3) items  $e$  and  $a$  co-occur 6 times in transactions.



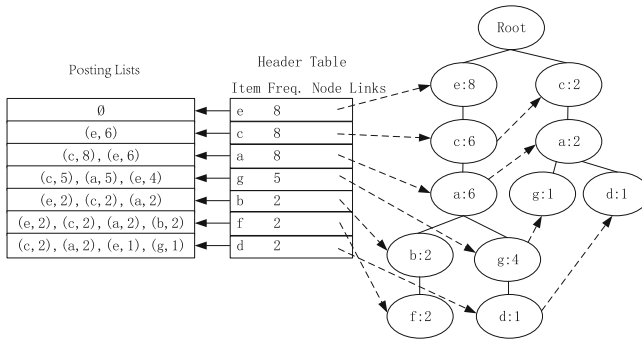


Fig. 2. Example FP\*-Tree

An FP\*-tree can be built by a procedure that is a minor modification of that for building the FP-tree. The construction of an FP-tree requires two scans of the transactional database. After the first scan, the header table is constructed. The posting list of each item can be built during the second scan when the tree structure is being constructed. For each transaction, the items are first sorted following the item order in the header table. Then the items are inserted into the tree structure one by one. In addition, each item  $i$  (except the first one) in the transaction is added to the posting lists of the items whose ranks are lower in the header table than that of  $i$ , and the corresponding counters are updated. The following example shows how to build posting lists during the second scan.

**Example 4.** Figure 3 shows how posting lists are updated as transactions are processed. So far, the header table has been constructed for  $\sigma = 2$  (Fig. 2). Initially, the posting list of each item is empty. When transaction  $T_1$  in Table 1 is processed, the items in  $T_1$  are re-ordered as  $e, c, a, b, f$  to follow the item order in the header table. Item  $o$  is removed, since it does not occur in the header table, meaning that its frequency is less than  $\sigma = 2$ . The posting lists of item  $c, a, b$ , and  $f$  are updated as shown in Fig. 3. Take item  $a$  as an example. Items  $e$  and  $c$  are added to its posting list, since their ranks in the header table are higher than that of  $a$ . The corresponding counters are set to 1 because both  $e$  and  $c$  co-occur with  $a$  in  $T_1$ . Similarly, when  $T_2$  is processed, the posting lists are updated as shown.

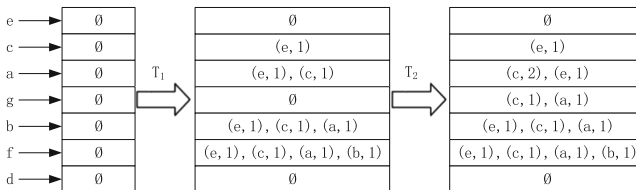


Fig. 3. Building posting lists

### 3.2 Algorithm

Given a transaction database  $\mathcal{D}$  and a category tree  $CT$ , a user defined frequency threshold  $\sigma$ , and the desired number of MDFIs  $k$ , the state-of-the-art method FPMAX [8] can compute the result  $k$ MDFIs by first discovering all maximal frequent itemsets (MFIs) using the FP-tree and then computing the diversity score of each MFI. Finally, the  $k$ MDFIs are the  $k$  MFIs with the largest diversity scores. The limitation of FPMAX is that it has to compute many MFIs with small diversity scores that do not contribute to the result.

The proposed bound-based algorithm adopts the FP\*-tree, making it possible to avoid computing MFIs with small diversity scores, thus saving substantial computational costs. Algorithm 1 shows the pseudo code. It first uses Algorithm 2 to construct a superset for each item and then uses Algorithm 3 to compute an upper bound on the diversity score for each item. Next, the items in the header table are sorted in non-increasing order of their bounds, and the algorithm then processes the items using this bound-based order. For each item  $i$ , the algorithm computes the MFIs containing  $i$  as does FPMAX. The discovered MFIs are added to the candidate set, and the diversity score of the current  $k^{th}$  MFI is recorded as  $\tau$ . Next, when an item is to be processed, its bound is first compared with  $\tau$ . If the bound is smaller than  $\tau$ , the algorithm returns the current top- $k$  MFIs; otherwise, the item is processed, and the MFIs containing the item are computed. Function `getNextItem()` follows the bound-based order in the header table and returns the next unprocessed item, and function `MFI( $i$ )` is the sub-routine in algorithm FPMAX that computes the MFIs that contain item  $i$ .

```

input : Transactional database  $\mathcal{D}$ , category tree  $CT$ , frequency threshold  $\sigma$ ,
        desired number of MFIs  $k$ 
output: Top- $k$  MDFIs
1  $\overline{X}_i \leftarrow$  call Algorithm 2 to construct a superset of the MFIs containing item  $i$  in
  the header table according to  $\sigma$ ;
2 Call Algorithm 3 to compute the upper bound  $b_i$  on the diversity scores of the
  MFIs containing item  $i$  using  $\overline{X}_i$ ;
3 Sort the items in the header table in descending order of their upper bounds;
4  $\tau \leftarrow -\infty$ ; ▷ The diversity score of the current  $k^{th}$  MFI.
5 while  $i \leftarrow \text{getNextItem}() \wedge b_i \geq \tau$  do
6    $\mathcal{X} \leftarrow \text{MFI}(i)$ ;
7   foreach  $X \in \mathcal{X}$  do
8     Compute the diversity score  $div(X)$  of  $X$ ;
9     Add  $X$  to the candidate set;
10     $\tau \leftarrow$  the diversity score of the current  $k^{th}$  MFI in the candidate set;
11  end
12 end
13 Return the top- $k$  MFIs in the candidate set;

```

**Algorithm 1.** Bound-based Algorithm

### 3.3 Bounds on Diversity Scores

To define bounds on diversity scores, we need the concept of the tail of an itemset.

**Definition 6.** *The tail  $t(X)$  of an itemset  $X$  is the item in  $X$  whose rank in the header table is lower than the ranks of all the other items in  $X$ .*

We derive a bound on the diversity score of an itemset (Lemma 2) and a bound on the diversity scores of the MFIs who have the same tail (Definition 6 and Lemma 4). Our bound-based algorithm efficiently computes the top- $k$  MDFIs using these bounds.

**Lemma 1.** *Given an FP\*-tree,  $\sigma$ , and item  $i$ , Algorithm 2 constructs a superset  $\overline{X}_i$  of all possible MFIs whose tails are  $i$ . Note that there may exist multiple MFIs whose tails are  $i$ .*

**Proof.** Let  $M_i$  be the set containing all MFIs  $X$  such that  $t(X) = i$ . We now prove that  $\overline{X}_i$  as constructed by Algorithm 2 is a superset of any  $X$  in  $M_i$ . Suppose an itemset  $X'$  exists in  $M_i$  that is not a subset of  $\overline{X}_i$ . Then there must be an item  $i'$  in  $X'$  that is not in  $\overline{X}_i$ . Since  $X' \in M_i$  is an MFI and  $t(X') = i$ , items  $i'$  and  $i$  must co-occur no fewer than  $\sigma$  times in transactions, and the rank of  $i'$  must be higher than that of  $i$  in the header table. According to the method for constructing  $L(i)$ , item  $i'$  must be contained in  $L(i)$ . And based on Algorithm 2, item  $i'$  should be added to  $\overline{X}_i$ , which contradicts the assumption that  $i'$  is not in  $\overline{X}_i$ . Thus, we have  $\forall X \in M_i (\overline{X}_i \supseteq X)$ .

```

input : FP*-tree, support  $\sigma$ , item  $i$ 
output: A superset of all possible MFIs whose tail is  $i$ 

1 Get the posting list  $L(i)$  of item  $i$  from the FP*-tree;
2 Add  $i$  to  $\overline{X}_i$ ;
3 foreach item  $i'$  in  $L(i)$  do
4   | if the count associated with  $i' \geq \sigma$  then
5   |   | Add  $i'$  to  $\overline{X}_i$ ;
6   | end
7 end

```

**Algorithm 2.** Superset Construction

**Lemma 2.** *Given an itemset  $X$ , the set  $X^w = \{i_L, i_R\}$  consists of the furthest apart pair of items in  $X$  according to the shortest path length in the category tree. Then, the diversity score of  $X^w$  is an upper bound on the diversity score of  $X$ , i.e.,  $div(X^w) \geq div(X)$ .*

*Proof.* Let  $C$  be the lowest common ancestor of the items in  $X$ , and let the level of  $C$  in the category tree be  $s$ . Then  $C$  is also the lowest common ancestor of the items in  $X^w$ , since set  $X^w = \{i_L, i_R\}$  consists of the furthest apart pair of items in  $X$  according to the shortest path length. Then, we have  $|GP(X, s)| = |GP(X^w, s)| = 1$ . For  $s + 1 \leq l \leq h - 1$ ,  $MF(X, l) \leq 1 = MF(X^w, l)$ . This holds because (i)  $X^w$  contains only two items, and  $C$  is the lowest common ancestor of the two items at level  $s$ , so that the cardinality of the generalized pattern of  $X^w$  at level  $s + 1 \leq l \leq h - 1$  is 1, and (ii)  $|GF(X, l)| \leq |GF(X, l + 1)|$ . Hence,  $div(X^w) \geq div(X)$ .

**Example 5.** To exemplify Lemma 2, consider Fig. 4 where  $X = \{a, c, e\}$ . The furthest apart pair of items in  $X$  are  $a$  and  $e$ , so  $X^w = \{a, e\}$ . Both  $X$  and  $X^w$  have the same lowest common ancestor  $C_4$  and its level is 2 in the category tree in Fig. 1. The diversity score of  $X^w$  is  $(1/6) \cdot 1 = 0.17$ , and the diversity score of  $X$  is  $(1/6) \cdot (1/2) = 0.08$ , so that  $div(X^w) > div(X)$ .

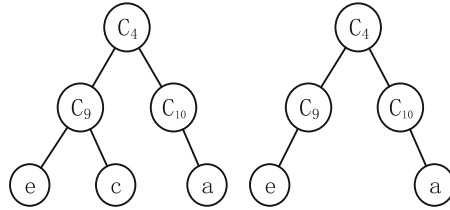


Fig. 4. Example of Lemma 2

**Lemma 3.** Let itemsets  $X_1$  and  $X_2$  each consist of two items, and let  $|GP(X_1, s_1)| = 1$  and  $|GP(X_2, s_2)| = 1$ . If  $s_1 \leq s_2$ ,  $div(X_1) \leq div(X_2)$ .

*Proof.* Since itemset  $X_1$  consists of two items and  $|GP(X_1, s_1)| = 1$ , we have  $div(X_1) = \sum_{l=h-1}^{s_1+1} PLF(l) \cdot 1$ . Similarly, we obtain  $div(X_2) = \sum_{l=h-1}^{s_2+1} PLF(l) \cdot 1$ . If  $s_1 \leq s_2$ , we have  $div(X_1) \leq div(X_2)$ .

**Lemma 4.**  $div(\overline{X}_i^w)$  is an upper bound on the diversity score of any MFI whose tail is  $i$ .

*Proof.* According to Lemma 1, set  $\overline{X}_i$  is a superset of any MFI whose tail is  $i$ . Let  $X$  represent any MFI whose tail is  $i$ . Sets  $\overline{X}_i^w$  and  $X^w$  consist of the furthest apart pair of items in  $\overline{X}_i$  and  $X$ , respectively. Given a category tree,  $|GP(\overline{X}_i^w, s_1)| = 1$  and  $|GP(X^w, s_2)| = 1$ . Since  $X \subseteq \overline{X}_i$ , it follows that  $s_1 \geq s_2$ . Based on Lemma 3,  $div(\overline{X}_i^w) \geq div(X^w)$ . According to Lemma 2, we have  $div(X^w) \geq div(X)$ . Hence, we derive  $div(\overline{X}_i^w) \geq div(X)$ , meaning that  $div(\overline{X}_i^w)$  is an upper bound on the diversity score of any MFI with tail  $i$ .

According to Lemma 4, the bound on the diversity score of any MFI whose tail is  $i$  is the diversity score of  $\overline{X}_i^w$  which consists of the furthest apart pair of items in  $\overline{X}_i$  returned by Algorithm 2. Actually, the diversity score of  $\overline{X}_i^w$  can be computed without explicitly finding the furthest apart pair of items in  $\overline{X}_i$  using Algorithm 3. It takes the codes of the items in  $\overline{X}_i$  as input. The length of each code is the height of the category tree. Each element in the code of an item corresponds to a node on the path from the root to the item. The diversity score bound is initialized to 0. The algorithm checks the elements at the same level in the codes of all the items from  $level = h - 1$  to the level of the lowest common ancestor of all the items. At each level, the corresponding proportional level factor is added to the diversity score bound. It is because that for  $\overline{X}_i^w$ , before reaching the level where the lowest common ancestor is found,  $MF(\overline{X}_i^w, l) = 1$ . Finally, the bound on the diversity score of any MFI with tail  $i$  is returned.

**Example 6.** Table 2 shows itemsets  $\overline{X}_i$  and the diversity score bounds  $div(\overline{X}_i^w)$  computed by Algorithms 2 and 3, given the FP\*-tree in Fig. 2. Then bound-based order of the items in the header table is  $g, b, f, a, d, c$ . Suppose the top-1 MFI is requested. The bound-based algorithm first computes the MFIs whose tail is item  $g$ , i.e., itemset  $\{a, c, e, g\}$  with diversity score 0.78. Now, it is found that the diversity score of the current top-1 candidate exceeds the bound of the item to be processed next ( $0.78 > 0.5$ ). The algorithm returns  $\{a, c, e, g\}$  as the top-1 result and terminates. If using the FPMAX algorithm, following the order of the item frequency before finding the top-1 result, items  $c, a, g, b$ , and  $f$  have to be processed. Even when item  $d$  has been processed, FPMAX is still not aware of whether the MFIs that have not been found will have higher diversity scores.

```

input : Codes of items in set  $\overline{X}_i$ 
output: Bound on the diversity score of the MFIs with tail  $i$ 
1  $div \leftarrow 0$ ;
2  $level \leftarrow h - 1$ ;
3 while  $level > 0$  do
4   if the elements at  $level$  in all the codes are the same then
5     | Break;
6   end
7   else
8     |  $div \leftarrow div + PLF(level)$ ;
9     |  $level \leftarrow level - 1$ ;
10  end
11 end
12 Return  $div$ ;

```

**Algorithm 3.** Bound Computation

**Table 2.** Example diversity score bounds

Item $i$	$\overline{X}_i$	$div(\overline{X}_i^w)$
$g$	$a, c, e, g$	1
$b$	$a, b, c, e$	0.5
$f$	$a, b, c, e, f$	0.5
$a$	$a, c, e$	0.17
$d$	$a, c, d$	0.17
$c$	$c, e$	0

## 4 Empirical Study

The proposed algorithms are evaluated on a real commercial data set that consists of 3,040,715 transactions. The number of unique items is 37,984. The height of the category tree is 5. The number of non-leaf nodes in the category tree is 1,947. All algorithms have been implemented using Java and performed on a machine with Intel(R) Core(TM) i5-4590 CPU @ 3.30 GHz 3.30 GHz, 16 GB RAM and the Windows 10 Professional operating system.

### 4.1 Result Investigation

**Maximal Diversified Frequent Itemsets (MDFIs).** Table 3 shows example MDFIs and the number of MDFIs found from the real data set using various frequency threshold  $\sigma$ , where  $N$  is the number of transactions in the data set. The items in the discovered MDFIs belong to different categories. Take MDFI “yoghurt, pear” as an example. Item “yoghurt” belongs to category “drink” and item “pear” belongs to category “vegetable”. The data set used only contains food-related categories. It is expected that more interesting MDFIs will be found if other types of items, such as clothes and home appliances, are included. When  $\sigma$  is small, e.g.,  $0.000005 \times N$ , a large number (980,241) of MDFIs are found.

**Table 3.** Maximal diversified frequent itemsets

$\sigma = 0.001 \times N$		$\sigma = 0.0005 \times N$		$\sigma = 0.0001 \times N$	
MDFIs: 71	$div$	MDFIs: 457	$div$	MDFIs: 11313	$div$
yoghurt, banana	1	yoghurt, salt	1	stationary, auto accessories	1
pork, tomato	1	pork, pumpkin	1	yoghurt, fish	1
rice, fish	1	tomato, shrimp	1	rice, chicken	1
$\sigma = 0.00005 \times N$		$\sigma = 0.00001 \times N$		$\sigma = 0.000005 \times N$	
MDFIs: 32990	$div$	MDFIs: 356204	$div$	MDFIs: 980241	$div$
yoghurt, hot dog	1	pistachio, tea	1	basket, chocolate, pistachio, sugar box, tea	1
yoghurt, dumpling	1	Coca Cola, nuts	1	yoghurt, shorts	1
potato, curry	1	oil, shampoo	1	sugar, crab	1

When  $\sigma$  is large, e.g.,  $0.001 \times N$ , only 71 MDFIs are discovered. In this dataset, no MDFI is found when  $\sigma \geq 0.005 \times N$ .

**Length of MDFIs.** Figure 5 shows the length distribution of the MDFIs. When  $\sigma$  is large ( $0.0005 \times N$ ), all MDFIs are of length 2. When  $\sigma = 0.00005 \times N$ , around 90% of the MDFIs are of length 2 and 10% of the MDFIs contain 3 items. When  $\sigma$  is small ( $0.000005 \times N$ ), 50% of the MDFIs are of length 2, 40% contain 3 items, and 10% are of length larger than 3. It is expected that longer MDFIs are found when using small  $\sigma$  values, since more items are considered as frequent.

**Diversity Score Distribution.** Figure 6 shows the diversity score distribution of the discovered MDFIs. When  $\sigma$  is large ( $0.0005 \times N$ ), 66% of the MDFIs have diversity scores from 0.75 to 1, and 10% have diversity scores from 0.5 to 0.75. When  $\sigma$  is small ( $0.000005 \times N$ ), 42% of the MDFIs have diversity scores from 0.75 to 1, 19% have diversity scores from 0.25 to 0.5, and 40% have diversity scores from 0 to 0.25. As expected, when using small  $\sigma$  values, long MDFIs with high diversity scores are found.

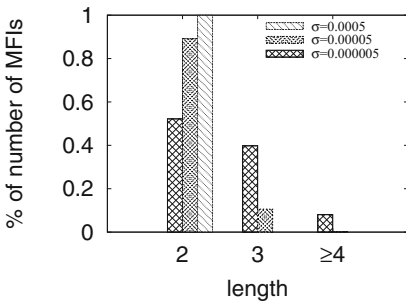


Fig. 5. Length distribution

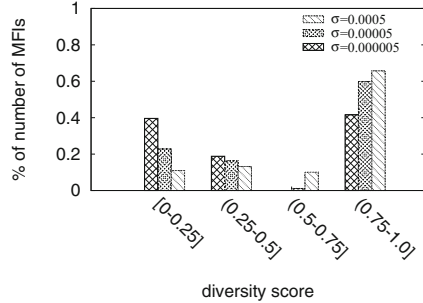


Fig. 6. Diversity distribution

### 4.2 Efficiency

We proceed to evaluate the performance of the proposed algorithm and a basic algorithm under different parameter settings.

**Basic Algorithm.** No algorithm exists that targets top- $k$  MDFIs. Instead, for comparison, we provide a basic algorithm that extends some existing techniques. The basic algorithm has two steps. It firstly finds all maximal frequent itemsets using the FPMAX [8] algorithm. Then, it computes the diversity score of each discovered maximal frequent itemset using the Item-Encoding algorithm [17]. Finally, the top- $k$  MDFIs with the highest diversity scores are returned.

**Varying the Number of Requested Sets  $k$ .** Recall that to obtain the top- $k$  MDFIs, the basic algorithm computes all the MFIs from the data set, while the bound-based algorithm only computes a few candidate MFIs. Figure 7 shows

the number of MFIs computed and the CPU time of the two algorithms when  $k$  is varied from 1 to 50. Parameter  $\sigma$  is set to 4000, which is roughly 0.13% of the transactions in the data set. Note that the number of MFIs computed and the CPU time of the basic algorithm do not change with  $k$ , since whatever  $k$  is, the basic algorithm always computes all the MFIs in the data set and ranks them. Nevertheless, the number of MFIs computed, and the CPU time of the bound-based algorithm increase as  $k$  increases. Because the more MDFIs that are requested, the more candidates are computed, yielding more computational cost. When  $k = 1$ , the number of MFIs computed using the bound-based algorithm is 55% less than that of the basic algorithm and the CPU time of the bound-based algorithm is only 0.6% of the CPU time of the basic algorithm. When  $k$  is set to 10 and 20, the bound-based algorithm also significantly outperforms the basic algorithm. When  $k = 50$ , the performance of the bound-based algorithm and the basic algorithm are the same. The reason is that there are 38 MFIs in the data set under the current parameter setting. Requesting top-50 MDFIs incurs the same computational cost for both algorithms.

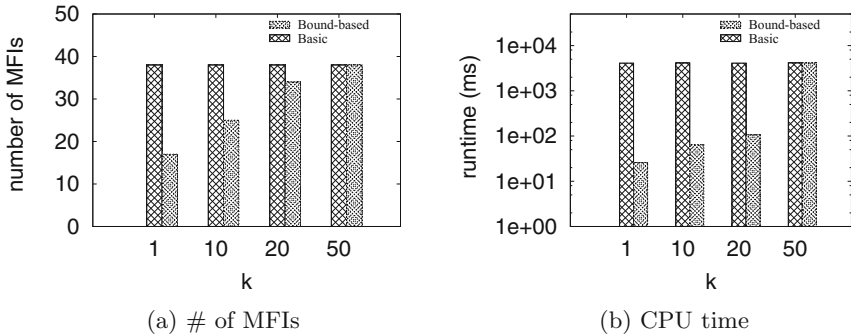


Fig. 7. Varying  $k$

**Varying the Frequency Threshold  $\sigma$ .** Figure 8 shows the number of MFIs computed and the CPU time of the two algorithms when  $\sigma$  is varied from 1500 (0.05% of the transactions in the data set) to 4000 (0.1% of the transactions in the data set). The number of requested MDFIs  $k$  is fixed at 10. When  $\sigma$  is small, many MFIs can be discovered. When  $\sigma$  is large, only few MFIs exist. Hence, as  $\sigma$  increases, the computational costs of both algorithms decrease. The bound-based algorithm beats the basic algorithm for all values of  $\sigma$ . The number of MFIs computed using the bound-based algorithm is 53%–83% of that using the basic algorithm. The CPU time of the bound-based algorithm is 0.6%–7.2% of the CPU time of the basic algorithm.

**Scalability.** To study how the computational cost of the proposed algorithm changes when varying the size of the data set, we have generated five data sets from the original data set by randomly selecting 200K, 400K, 600K, 800K, and



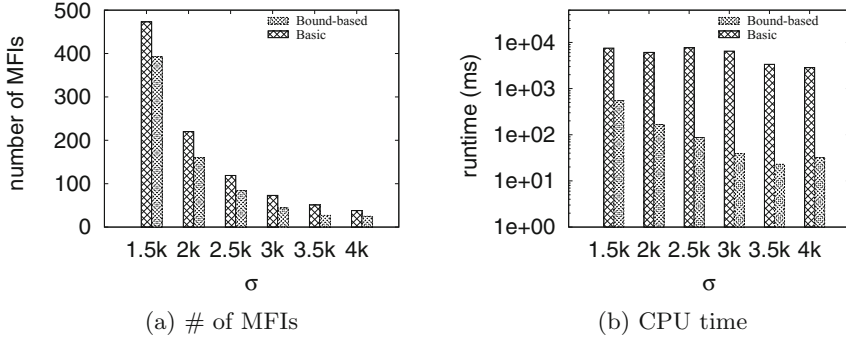


Fig. 8. Varying  $\sigma$

1M transactions. Figure 9 shows the number of MFIs computed and the CPU time of the two algorithms when the size of the data set is varied. Parameter  $\sigma$  is set to 500, which is roughly 0.25%, 0.125%, 0.083%, 0.063%, and 0.05% of the number of transactions in the five data sets, respectively. The number of requested MDFIs  $k$  is fixed at 1. The computational costs of both the basic and the bound-based algorithms increase as the size of data set increases. On the five data sets, the bound-based algorithm outperforms the basic algorithm by orders of magnitude in terms of CPU time. The number of computed MFIs of the bound-based algorithm is 33.3%–82.3% of that of the basic algorithm.

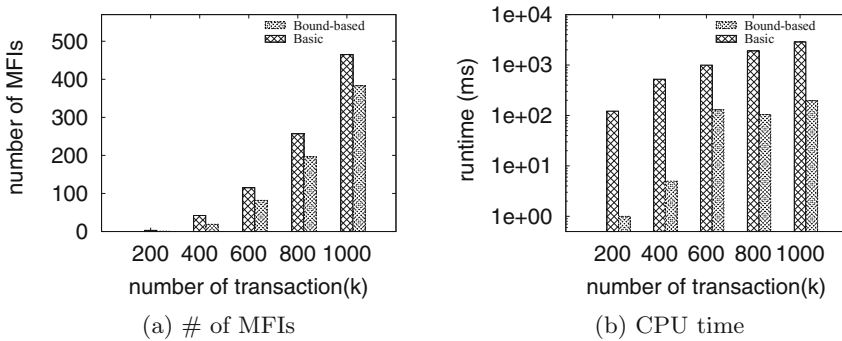


Fig. 9. Varying the number of transactions

## 5 Related Work

**Frequent Itemsets with Various Constraints.** Since mining frequent itemsets from transactional databases involves an exponential mining space and generates a huge number of itemsets, efficient discovery of constrained or user-interest based frequent itemsets is attractive. In many real-world scenarios, it is

often sufficient to mine a small and interesting representative of frequent itemsets. Hence, various constraints have been posed on the frequent itemsets in the literature. An itemset  $X$  is closed in a data set if there exists no superset that has the same frequency as  $X$ . Pasquier et al. [13] propose the A-Close algorithm that uses a closure mechanism to find frequent closed itemsets. A frequent itemset is maximal if none of its supersets are frequent. Burdick et al. [4] introduce the MAFIA algorithm for mining maximal frequent itemsets from a transactional database. A frequent itemset is periodic-frequent if it appears at a user-specified regular interval in the database. Tanbeer et al. [19] present the periodic-frequent pattern tree that captures the database contents in a highly compact manner and enables a pattern growth mining technique to generate the complete set of periodic-frequent itemsets in a database for user-given periodicity and support thresholds. Top- $k$  frequent itemset mining finds interesting itemsets from the highest support to the  $k$ -th support. The CRM and CRMN algorithms [14] are proposed to mine top- $k$  frequent itemsets efficiently. In utility mining, each item has external utility such as a profit or price and internal utility that refers to a non-binary value in a transaction. The importance of an itemset is measured by the concept of utility, which is the sum of the products of external and internal utilities of items in the itemset. An itemset is called a high utility itemset [15] when its utility is no less than a user-specified minimum utility threshold. Mallick et al. [12] consider the problem of the incremental mining of sequential patterns when new transactions or new customers are added to an original database. They present an algorithm for mining frequent sequences that uses information collected during an earlier mining process to cut down the cost of finding new sequential patterns in the updated database. Frequent weighted itemset mining considers a database where each item in a transaction may have a different significance. Vo et al. [20] propose a method for mining frequent weighted itemsets using WIT-trees. The diverse frequent itemset mining [17] uses the DiverseRank to rank the frequent itemsets based on the items' categories. Later, Swamy et al. [18] study the diverse frequent itemsets in the context where there concept hierarchies are unbalanced.

In this paper, we study the MDFI that extends the diverse frequent itemset by considering the maximality constraint. And an efficient bound-based algorithm is proposed for mining the top- $k$  MDFIs.

**Maximal Frequent Itemset Mining.** MAFIA [5] mines maximal frequent itemsets (MFIs) using a depth-first traversal of the itemset lattice with pruning mechanisms and combining a vertical bitmap representation of the database. GenMax [6,7] is a backtrack search based algorithm for mining MFIs. It uses progressive focusing to perform maximality checking, and it uses diffset propagation to perform fast frequency computation. MinMax [21] is also based on depth-first traversal and iterations for mining MFIs. It removes all the non-maximal frequent itemsets without enumerating all the frequent itemsets from smaller ones. It backtracks to the proper ancestor directly, instead of level by level. Algorithms LFIMiner and LFIMiner-ALL [10] adopt a pattern fragment growth methodology based on the FP-tree for mining maximum length frequent

itemsets that is a subset of the MFIs. Yang [23] studied the complexity-theoretic aspects of MFI mining, from the perspective of counting the number of solutions. MaxDomino [16] uses the notions of dominance factor and collapsibility of transaction for efficiently mining MFIs. It employs a top-down strategy with selective bottom-up search. Pincer-Search [11] combines both bottom-up and top-down search for discovering MFIs. A restricted search is conducted in the top-down direction for maintaining and updating the maximum frequent candidate set, which is used for early pruning of candidates that would normally be encountered in the bottom-up search. CfpMfi [22] is a depth-first search algorithm based on CFP-tree for mining MFIs. It uses a variety pruning techniques and an item ordering policy to reduce the search space. DepthProject [1] finds long itemsets using a depth first search of a lexicographic tree of itemsets, and it uses a counting method based on transaction projections along its branches. MaxMiner [3] employs a breadth-first traversal of the search space and reduces database scanning by employing a look-ahead pruning strategy, i.e., if a node with all its extensions can be determined to be frequent, there is no need to further process that node. FPMAX [8] is an extension of the well know FP-growth [9] for mining MFIs. The maximal frequent itemset tree (MFI-tree) is used to keep track of all maximal frequent itemsets.

The basic algorithm for mining MDIFs proposed in this paper uses the state-of-the-art FPMAX as a component. And the proposed bound-based algorithm outperforms the basic algorithm significantly.

## 6 Conclusions

This work studies the problem of finding the top- $k$  most diverse itemsets that are frequent. It tries to find long frequent itemsets of items belonging to different categories. Since no existing algorithm targets top- $k$  MDIFs mining, we propose a basic algorithm that extends existing techniques. However, the basic algorithm fails to scale well to large data sets. We also propose the so-called FP\*-tree along with a bound-based algorithm that is able to reduce the computational costs very significantly. Extensive experiments conducted on a large data set demonstrate that the proposed method consistently outperforms the basic algorithm.

## References

1. Agarwal, R.C., Aggarwal, C.C., Prasad, V.V.V.: Depth first generation of long patterns. In: KDD, pp. 108–118 (2000)
2. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: SIGMOD, pp. 207–216 (1993)
3. Bayardo Jr., R.J.: Efficiently mining long patterns from databases. SIGMOD Rec. **27**(2), 85–93 (1998)
4. Burdick, D., Calimlim, M., Flannick, J., Gehrke, J., Yiu, T.: MAFIA: a maximal frequent itemset algorithm. IEEE Trans. Knowl. Data Eng. **17**(11), 1490–1504 (2005)

5. Burdick, D., Calimlim, M., Gehrke, J.: MAFIA: a maximal frequent itemset algorithm for transactional databases. In: ICDE, pp. 443–452 (2001)
6. Gouda, K., Zaki, M.J.: GenMax: an efficient algorithm for mining maximal frequent itemsets. *Data Min. Knowl. Discov.* **11**(3), 223–242 (2005)
7. Gouda, K., Zaki, M.J.: Efficiently mining maximal frequent itemsets. In: ICDM, pp. 163–170 (2001)
8. Grahne, G., Zhu, J.: High performance mining of maximal frequent itemsets. In: HPDM (2003)
9. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: SIGMOD, pp. 1–12 (2000)
10. Hu, T., Sung, S.Y., Xiong, H., Fu, Q.: Discovery of maximum length frequent itemsets. *Inf. Sci.* **178**(1), 69–87 (2008)
11. Lin, D.I., Kedem, Z.M.: Pincer-search: an efficient algorithm for discovering the maximum frequent set. *IEEE Trans. Knowl. Data Eng.* **14**(3), 553–566 (2002)
12. Mallick, B., Garg, D., Grover, P.S.: Incremental mining of sequential patterns: progress and challenges. *Intell. Data Anal.* **17**(3), 507–530 (2013)
13. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Discovering frequent closed itemsets for association rules. In: Beeri, C., Buneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 398–416. Springer, Heidelberg (1999). [https://doi.org/10.1007/3-540-49257-7\\_25](https://doi.org/10.1007/3-540-49257-7_25)
14. Pyun, G., Yun, U.: Mining top-k frequent patterns with combination reducing techniques. *Appl. Intell.* **41**(1), 76–98 (2014)
15. Ryang, H., Yun, U., Ryu, K.H.: Fast algorithm for high utility pattern mining with the sum of item quantities. *Intell. Data Anal.* **20**(2), 395–415 (2016)
16. Srikumar, K., Bhasker, B.: Efficiently mining maximal frequent sets in dense databases for discovering association rules. *Intell. Data Anal.* **8**(2), 171–182 (2004)
17. Srivastava, S., Kiran, R.U., Reddy, P.K.: Discovering diverse-frequent patterns in transactional databases. In: COMAD, pp. 69–78 (2011)
18. Kumara Swamy, M., Reddy, P.K., Srivastava, S.: Extracting diverse patterns with unbalanced concept hierarchy. In: Tseng, V.S., Ho, T.B., Zhou, Z.-H., Chen, A.L.P., Kao, H.-Y. (eds.) PAKDD 2014. LNCS (LNAI), vol. 8443, pp. 15–27. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-06608-0\\_2](https://doi.org/10.1007/978-3-319-06608-0_2)
19. Tanbeer, S.K., Ahmed, C.F., Jeong, B.-S., Lee, Y.-K.: Discovering periodic-frequent patterns in transactional databases. In: Theeramunkong, T., Kijssirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS (LNAI), vol. 5476, pp. 242–253. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-01307-2\\_24](https://doi.org/10.1007/978-3-642-01307-2_24)
20. Vo, B., Coenen, F., Le, B.: A new method for mining frequent weighted itemsets based on WIT-trees. *Expert Syst. Appl.* **40**(4), 1256–1264 (2013)
21. Wang, H., Li, Q., Ma, C., Li, K.: A maximal frequent itemset algorithm. In: RSFD-GrC, pp. 484–490 (2003)
22. Yan, Y., Li, Z., Wang, T., Chen, Y., Chen, H.: Mining maximal frequent itemsets using combined FP-tree. In: Webb, G.I., Yu, X. (eds.) AI 2004. LNCS (LNAI), vol. 3339, pp. 475–487. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-30549-1\\_42](https://doi.org/10.1007/978-3-540-30549-1_42)
23. Yang, G.: The complexity of mining maximal frequent itemsets and maximal frequent patterns. In: KDD, pp. 344–353 (2004)

# **Privacy and Graph**



# Efficient Local Search for Minimum Dominating Sets in Large Graphs

Yi Fan<sup>1,2,3</sup>, Yongxuan Lai<sup>1</sup>(✉), Chengqian Li<sup>4</sup>, Nan Li<sup>5</sup>, Zongjie Ma<sup>3</sup>,  
Jun Zhou<sup>3</sup>, Longin Jan Latecki<sup>5</sup>, and Kaile Su<sup>3</sup>

<sup>1</sup> Shenzhen Research Institute/Software School, Xiamen University, Xiamen, China

yifan.sysu@gmail.com, laiyongxuan@gmail.com

<sup>2</sup> Guangxi Key Lab of Trusted Software,

Guilin University of Electronic Technology, Guilin, China

<sup>3</sup> Griffith University, Brisbane, Australia

zongjie.ma@griffithuni.edu.au, {jun.zhou,k.su}@griffith.edu.au

<sup>4</sup> NetEase Guangzhou AI Lab, Guangzhou, China

lcqn2776@corp.netease.com

<sup>5</sup> Temple University, Philadelphia, USA

{nan.li,latecki}@temple.edu

**Abstract.** The Minimum Dominating Set (MinDS) problem is an NP-hard problem of great importance in both theories and applications. In this paper, we propose a new local search algorithm ScBppw (Score Checking and Best-picking with Probabilistic Walk) to solve the MinDS problem in large graphs. For diversifying the search, our algorithm exploits a tabu strategy, called Score Checking (SC), which forbids a vertex to be added into the current candidate solution if the vertex's score has not been changed since the last time it was removed out of the candidate solution. Also, to keep a good balance between intensification and diversification during the search, we propose a strategy that combines, in a novel way, best-picking with probabilistic walk at removing stages. At this stage, the algorithm selects a vertex with the minimum loss, or other vertices in the candidate solution with a probability proportional to the their degrees, depending on how repeatedly the area has been visited. Experimental results show that our solver significantly outperforms state-of-the-art MinDS solvers. Also we conducted several experiments to show the individual impacts of our novelties.

**Keywords:** Minimum dominating set · Score-based partition · Best-picking

## 1 Introduction

Given a simple undirected graph  $G = (V, E)$ , a dominating set is a subset  $D \subseteq V$  s.t. every vertex outside  $D$  has at least one neighbor in  $D$ . The Minimum

Supported by the Natural Science Foundation of China (61672441, 61872154), the Shenzhen Basic Research Program (JCYJ20170818141325209), and the NSF grants (IIS-1814745, IIS-1302164).

© Springer Nature Switzerland AG 2019

G. Li et al. (Eds.): DASFAA 2019, LNCS 11447, pp. 211–228, 2019.

[https://doi.org/10.1007/978-3-030-18579-4\\_13](https://doi.org/10.1007/978-3-030-18579-4_13)

Dominating Set (MinDS) problem is to find a dominating set of the minimum size. MinDS arises in many application areas, *e.g.*, biological networks [20], metro networks [1], power networks [13], computer vision [21, 26], multi-document summarization [23], and wireless communication [22]. Among various algorithms, *e.g.* [6, 10], local search is an effective approach and it is powerful on difficult graphs.

### 1.1 Local Search Techniques

Local search often suffers from the cycling problem, *i.e.*, the search may spend too much time visiting a small part of the search space, thus various tabu strategies have been proposed to deal with this problem. Recently [5] proposed the so-called configuration checking (CC) strategy, and after that numerous variants of the CC strategy have been adopted to solve a wide range of combinatorial problems *e.g.*, [11, 24, 25]. The idea of the CC strategy can be described as follows. If a vertex is removed out of the candidate solution, then it is forbidden to be added back until its configuration is changed, *i.e.*, some neighboring vertex is added or removed out of the candidate solution.

The forbidding strength of CC can be too weak, which may lead a CC-based local search being stuck in a cycle. To escape from such a cycle, a CC-based local search usually needs some other diversifying strategies like constraint weighting [15], which is unfortunately time-consuming and impractical for large graphs. For diversifying the search, we exploit a strong CC-like tabu strategy, called Score Checking (SC), which forbids a vertex to be added into the current candidate solution if its score has not been changed since the last time it was removed out of the candidate solution. In this strategy, when a vertex is added or removed, some vertices in the neighborhood are released, while some others are still forbidden. This is different from usual CC variants.

To keep a good balance between intensification and diversification, we propose a strategy that combines, in a novel way, best-picking with probabilistic walk at removing stages. Best-picking with random walk has proved to be effective in solving the minimum vertex cover problem in large graphs [18]. More specifically, in greedy mode it chooses small-degree vertices while in random mode it selects a vertex from a uniform distribution. However, the strategy may still focus too much on small-degree vertices, in other words, there may not be enough chances to select a big-degree vertex. Therefore we adopt a probabilistic distribution to remove vertices in random mode. To be specific, a vertex is selected to be removed with a probability proportional to its degree. Evidently, this strategy gives more chances to big-degree vertices.

Also, when we are doing local search in large instances, the complexity becomes a big issue. In each step there can be millions of possible moves, and thus it is difficult to obtain a local move which maximizes certain kinds of benefits. Since most real-world graphs are sparse [3, 8, 10], it is beneficial to develop data structures so that the complexity of a single search step relies on the average degree rather than the number of vertices. For instance, a local search solver LMY-GRS [11] follows this idea and finds good solutions for the maximum weight clique problem. Inspired by this, we propose an efficient data structure named

score-based partition to implement best-picking. The score-based partition effectively cuts off the average complexity of exchanging vertices in the local search algorithm.

## 1.2 Our Contributions

In this paper, we develop a local search solver named ScBppw (Score Checking and Best-picking with Probabilistic Walk) based on the strategies above. For showing the effectiveness, we compare ScBppw with FastWMDS [24] on large graphs<sup>1</sup>. Experimental results show that (1) our solver ScBppw significantly outperforms FastWMDS on large graphs; (2) our proposed strategies play an important role in our solver.

## 2 Preliminaries

### 2.1 Basic Notations

We use  $N(v) = \{u \in V \mid \{u, v\} \in E\}$  to denote the set of  $v$ 's neighbours, and we use  $N[v]$  to denote  $N(v) \cup \{v\}$ . The degree of a vertex  $v$ , denoted by  $d(v)$ , is defined as  $|N(v)|$ . We use

$$N^2(v) = (N(v) \cup \{u \mid \{u, w\} \in E \text{ and } w \in N(v)\}) \setminus \{v\}$$

to denote the set of vertices whose distance from  $v$  is at most 2. Also we use  $\bar{d}_2(G)$  to denote the average size of  $N^2(v)$  over all the vertices, *i.e.*,  $\bar{d}_2(G) = \frac{1}{|V|} \sum_{v \in V} |N^2(v)|$ , suppressing  $G$  if understood from the context. In graph theory, we have the following proposition that is useful in implementing our solver.

**Proposition 1.**  $\sum_{v \in V} d(v) = 2|E|$ .

A vertex is said to be covered by a set  $D$  if it is in  $D$  or at least one of its neighbors is in  $D$ . Otherwise it is said to be uncovered by  $D$ . If  $u$ 's removal from  $D$  makes  $v$  become uncovered, we also say that  $u$ 's removal uncovers  $v$ . Likely if  $u$ 's addition into  $D$  makes  $v$  become covered, we also say that  $u$ 's addition covers  $v$ . For a vertex  $v \in D$ , the *loss* of  $v$ , denoted as  $loss(v)$ , is defined as the number of covered vertices that will become uncovered by removing  $v$  from  $D$ . For a vertex  $v \notin D$ , the *gain* of  $v$ , denoted as  $gain(v)$ , is defined as the number of uncovered vertices that will become covered by adding  $v$  into  $D$ . Both *gain* and *loss* are scoring properties. Obviously we have

**Proposition 2.** For all  $v \in V$ ,  $gain(v) \in [0, d(v) + 1]$ ,  $loss(v) \in [0, d(v) + 1]$ .

In MinDS solving, we have a proposition below which shows the set of vertices whose score needs to be updated.

<sup>1</sup> <http://networkrepository.com/networks.php>.



---

**Algorithm 1:** LocalSearchForMinDS( $G, cutoff$ )

---

**input** : A graph  $G = (V, E)$ , the *cutoff* time  
**output**: A dominating set of  $G$

```

1  $D \leftarrow \text{InitDS}(G)$ ;
2 while elapsed time < cutoff do
3   if  $D$  covers all vertices then
4      $D^* \leftarrow D$ ;
5     remove a vertex from  $D$  with the minimum loss, breaking ties randomly;
6     lastStepImproved  $\leftarrow$  false;
7   else
8     ExchangeVertices( $G, D$ );
9 return  $D^*$ ;
```

---

- Proposition 3.** 1. When a vertex  $u$  is added or removed, for any vertex  $v \notin N^2(u)$ ,  $gain(v)/loss(v)$  remains unchanged.  
2. If  $gain(v)$  or  $loss(v)$  is changed, then at least one vertex  $u \in N^2(v)$  has been added or removed.

In any step, a vertex  $v$  has two possible states: inside  $D$  and outside  $D$ . We use  $age(v)$  to denote the number of steps that have been performed since last time  $v$ 's state was changed.

## 2.2 An Iterative Local Search Framework

Algorithm 1 presents a local search framework for MinDS. It consists of the construction phase (Line 1) and the local search phase (Lines 2 to 8).

In our algorithm, we will adopt a simple greedy strategy to implement  $\text{InitDS}(G)$ , which works as follows. *Given an empty set  $D$ , repeat the following operations until  $D$  becomes a dominating set: select a vertex  $v \notin D$  with the maximum gain and add  $v$  into  $D$ , breaking ties randomly.* Actually we can use the data reduction rules in [2] to generate better initial solutions and more importantly simplify the input graph. Yet currently we do not do so, because our main concern in this paper is to develop an efficient local search algorithm.

In the local search phase, each time a  $k$ -sized dominating set is found (Line 3), the algorithm removes a vertex from  $D$  (Line 5) and proceeds to search for a  $(k - 1)$ -sized dominating set, until a certain time limit is reached (Line 2). A local move consists of exchanging a pair of vertices (Line 8): a vertex  $u \in D$  is removed from  $D$ , and a vertex  $v \notin D$  is added into  $D$ . Such an exchanging procedure is also called a step or an iteration by convention. In our algorithm, we also employ a predicate *lastStepImproved* s.t. *lastStepImproved* = *true* iff the number of uncovered vertices was decreased in the last step. This predicate will be used in the best-picking with probabilistic walk strategy we propose. Lastly, when the algorithm terminates, it outputs the smallest dominating set that has been found.

### 2.3 A Fast Hashing Function

In order to detect revisiting, we will use the hash function in [12] which is shown below.

**Definition 1.** *Given a candidate set  $D$  and a prime number  $p$ , we define the hash value of  $D$ , denoted by  $\text{hash}(D)$ , as  $(\sum_{v_i \in D} 2^i) \bmod p$ , which maps a candidate set  $D$  to its hash entry  $\text{hash}(D)$ .*

Also, they showed that each time a vertex is added or removed, the hash value can be updated in  $O(1)$  complexity.

### 2.4 Geometric Distribution

To analyze the complexity of our algorithm, we first introduce Geometric Distribution which depicts the probability that the first success occurs in a particular trial.

**Definition 2.** *A random variable  $X$  has a geometric distribution if the probability that the  $k^{\text{th}}$  trial (out of  $k$  trials) is the first success is  $\Pr(X = k) = (1 - p)^{k-1}p$ , where  $k$  is a positive integer and  $p$  denotes the probability of success in a single trial.*

Then the average number of trials needed for the first success is  $\frac{1}{p}$  according to the following theorem [19].

**Theorem 1.** *If  $X$  is a geometric random variable with parameter  $p$ , then the expected value  $E(X)$  is given by  $\frac{1}{p}$ .*

### 2.5 Sorting Vertices wrt. Degrees

Our algorithm implementation requires sorting vertices into non-decreasing order wrt. their degrees in advance, so we introduce an efficient sorting algorithm as follows. Considering that  $0 \leq d(v) \leq |V|$  for any  $v \in V$ , this satisfies the assumption of counting sort which runs in linear time [9]. So we have

**Proposition 4.** *Sorting the vertices into non-decreasing order wrt. their degrees can be done in  $O(|V|)$  complexity.*

### 2.6 Configuration Checking

In MinDS solving, there are three different CC strategies, *i.e.*, neighboring vertices based CC (NVCC) [5], two-level CC (CC<sup>2</sup>) [25] and three-valued two-level CC (CC<sup>2</sup>V3). In NVCC, the configuration of a vertex  $v$  refers to the state of the vertices in  $N(v)$ . On the other hand, in CC<sup>2</sup>, the configuration of a vertex  $v$  refers to the state of the vertices in  $N^2(v)$ .

We use  $V_{NVCC}$  to denote the set of vertices whose configuration is changed according to the NVCC strategy, and use  $V_{CC^2}$  to denote the set of vertices whose configuration is changed according to the CC<sup>2</sup> strategy. [25] presented the proposition below which shows that the forbidding strength of NVCC is stronger than that of CC<sup>2</sup>.

**Proposition 5.** *If  $V_{NVCC} = V_{CC^2}$  in some step, then  $V_{NVCC} \subseteq V_{CC^2}$  in the next step.*

### 3 Score Checking

We first define the notion of reversing operations and then propose a tabu strategy which is based on score change.

**Effects of a Local Move.** Given a vertex  $v$ , there are two possible operations: removing  $v$  from  $D$  and adding  $v$  into  $D$ , and we call them a pair of *reversing operations*. Adding a vertex into  $D$  has two types of effects: (1) turning some vertices from being uncovered to being covered by 1 vertex; (2) turning some vertices from being covered by  $c$  ( $c \geq 1$ ) vertices to being covered by  $(c + 1)$  vertices. Likely, a removing operation has analogous effects. Given a pair of reversing operations in a sequence of local search steps, we say that their effects are *neutralized* if and only if they have completely opposite effects.

Obviously if a pair of reversing operations occur consecutively, their effects will be neutralized. This is the worst case and any tabu strategy prevents such cases from happening. However, even though two reversing operations are not consecutive, their effects may still be neutralized. So the question is in under what condition their effects will not be neutralized.

**Our Tabu Strategy.** In fact it is impractical to record all the effects of an operation, since it requires much space and time for checking. To make it practicable, a compromising method is to memorize some important effects like *the set of vertices which become covered (or uncovered)*. Then we avoid any two reversing operations which cover and uncover the same set of vertices.

However, maintaining these sets and checking equality relation still consumes too much time. Therefore we choose to consider the score (gain or loss) of the operations. Then we attempt to avoid any pair of reversing operations if their scores are opposite. Now we give an example to describe our motivation.

*Example 1.* Suppose a vertex  $v$  is removed from the candidate solution at Step  $i$  with  $loss_i(v)$ , and then added back at Step  $j$  with  $gain_j(v)$ . We observe that

1. if  $loss_i(v) \neq gain_j(v)$ , then the effects of these two operations will not be neutralized;
2. even though  $loss_i(v) = gain_j(v)$ , their effects are not necessarily neutralized, since they may cover and uncover different sets of vertices respectively;
3. if  $gain_j(v)$  keeps unchanged after  $v$ 's removal, then  $v$ 's removal and later addition will uncover and cover the same set of vertices respectively.

Based on these observations, we propose a tabu strategy which is based on score change: *After a vertex is removed from the candidate solution, it cannot be added back until its score has been changed.* Throughout this paper, this strategy will be called Score Checking (SC).

Our strategy is implemented with a Boolean array named *free* s.t.  $free(v) = 1$  means  $v$  is allowed to be added into the candidate solution, and  $free(v) = 0$  otherwise. Then we maintain the *free* array as follows:

1. initially  $free(v)$  is set to 1 for all  $v \in V$ ;
2. when  $v$  is removed from  $D$ ,  $free(v)$  is set to 0;
3. when  $gain(v)$  is changed,  $free(v)$  is set to 1.

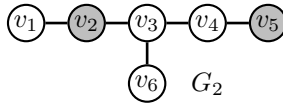
We notice that there is a concept called *promising variable* in SAT [17], which allows a variable to be flipped if its score becomes positive because of the flips of its neighboring variables. This concept is in some sense similar to the score checking strategy here.

### 3.1 Comparing SC to NVCC

Example 2 shows that SC has stronger forbidding strength than NVCC, while Example 3 shows the opposite.

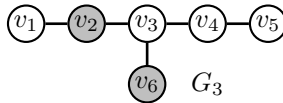
*Example 2.* Suppose that we have a graph  $G_2$  as below, the current candidate solution  $D = \{v_2, v_5\}$ ,  $free(v_3) = 1$  and  $free(v_4) = 0$ .

Now we add  $v_3$  into  $D$ . According to NVCC,  $free(v_4)$  will become 1, because one of its neighbors,  $v_3$ , has changed its state. However, according to SC,  $free(v_4)$  will still be 0, because  $gain(v_4)$  has not changed. So the forbidding strength of SC is stronger than that of NVCC in this case.



*Example 3.* Suppose that we have a graph  $G_3$  as below, the current candidate solution  $D = \{v_2, v_6\}$ ,  $free(v_3) = 1$  and  $free(v_4) = free(v_5) = 0$ .

Now we add  $v_3$  into  $D$ . According to NVCC,  $free(v_5)$  will still be 0, because none of its neighbors has changed its state. However, according to SC,  $free(v_5)$  will become 1, because  $gain(v_5)$  is decreased by 1. So the forbidding strength of NVCC is stronger than that of SC in this case.



Based on the examples above, we have

**Proposition 6.** *There exists a case in which the forbidding strength of SC is stronger than that of NVCC, and there also exists a case in which the forbidding strength of NVCC is stronger than that of SC.*

### 3.2 Comparing SC to $CC^2$

By Item 2 in Proposition 3, if a vertex's score is changed, then it must be that its configuration is changed according to  $CC^2$  strategy. Therefore if a vertex is released by SC, it must also be released by  $CC^2$ . So the forbidding strength of SC is at least as strong as that of  $CC^2$ . On the other hand, Example 2 shows that the forbidding strength of SC can sometimes be stronger than that of NVCC, and thus also stronger than that of  $CC^2$  by Proposition 5. Therefore we have a proposition below which shows that the forbidding strength of SC is strictly stronger than that of  $CC^2$ .

**Proposition 7.** *Let  $V_{SC}$  be the set of vertices which are free according to SC, then we have (1)  $V_{SC} \subseteq V_{CC^2}$  always hold; (2)  $V_{CC^2} \subseteq V_{SC}$  does not always hold.*

## 4 Our Algorithm for Exchanging Vertices

Here, we present the `ExchangeVertices( $G, D$ )` procedure in Algorithm 2, which adopts our new tabu strategy. In this algorithm, we use `uncov_v_num1` and

---

### Algorithm 2: ExchangeVertices

---

```

1  uncov_v_num1  $\leftarrow$  the number of uncovered vertices;
2  if lastStepImproved then
3      if hash(D) is marked then
4           $u \leftarrow$  a vertex in  $D$  with a probability proportional to its
              degree; // random mode
5      else
6          mark hash(D);
7           $u \leftarrow$  a vertex in  $D$  with the minimum loss, breaking ties
              randomly; // greedy mode
8  else
9       $u \leftarrow$  a vertex in  $D$  with the minimum loss, breaking ties
          randomly; // greedy mode
10 remove  $u$  from  $D$ ;
11 free(u)  $\leftarrow$  0;
12 free(x)  $\leftarrow$  1 for all  $x$  s.t. gain(x) is changed;
13  $v \leftarrow$  a random uncovered vertex;
14  $w \leftarrow$  a vertex  $u$  in  $N[v]$  s.t. free(u) = 1 with the greatest gain, breaking ties in
    favor of the greatest age;
15 add  $w$  into  $D$ ;
16 free(y)  $\leftarrow$  1 for all  $y$  s.t. gain(y) is changed;
17 uncov_v_num2  $\leftarrow$  the number of uncovered vertices;
18 if uncov_v_num2 < uncov_v_num1 then
19     lastStepImproved  $\leftarrow$  true;
20 else
21     lastStepImproved  $\leftarrow$  false;

```

---

$uncov\_v\_num_2$  to denote the number of uncovered vertices before and after exchanging vertices respectively. Furthermore, it adopts the two-stage exchange strategy in NuMVC [4].

#### 4.1 Removing Stage

In the removing stage (Lines 2 to 12), we propose a heuristic called *best-picking with probabilistic walk*. The intuition is as follows: if the local search has spent only a little time in the current area, we prefer greedy mode to find a good solution; otherwise we favor random mode to leave.

To be specific, we exploit the hash function in the preliminary section to approximately detect revisiting. Like [12], we set the prime number to be  $10^9 + 7$  and do not resolve collisions. Since in our experiments, our solver performs less than  $5 \times 10^8$  steps in any run, given the  $10^9 + 7$  hash entries, the number of collisions is negligible. Notice that we mark and detect the hash table only when  $lastStepImproved = true$ . The intuition is that if a solution is revisited together with the same  $lastStepImproved$  value as before, then the current area has been visited to a significant extent. In this situation, a vertex is removed from the candidate solution with a probability proportional to its degree. We tend to remove large-degree vertices, because we rarely remove them in the greedy mode, and this will further strengthen the diversification in our algorithm.

To implement probabilistic selections, we employ an array with length  $2|E|$ , where  $|E|$  is the number of edges, to store copies of the vertices. Before doing local search, given any vertex, say  $u$ , we put  $d(u)$  copies of  $u$  into the array. According to Proposition 1, there are exactly  $2|E|$  copies of the vertices in the array. When we perform the probabilistic selection, we do the following: *Randomly select an item in the array and if the result is in  $D$ , then return; otherwise repeat the random selection.*

Now we analyze the complexity of this procedure. Each time we obtain an item, the probability that it is in  $D$ , denoted by  $p$ , is at least  $\frac{|D|}{2|E|}$  (we use “at least” because usually a vertex in  $D$  has more than one copy in the array). Then by Theorem 1, the averaged number of trials needed for the first success is  $\frac{1}{p} \leq \frac{2|E|}{|D|}$ . Moreover, we find that  $\frac{2|E|}{|D|} \leq 500$  in most graphs. So the time complexity is  $O(\frac{2|E|}{|D|}) = O(1)$ .

#### 4.2 Adding Stage

In the adding stage (Lines 13 to 16), our algorithm chooses a random uncovered vertex and selects a vertex in its neighborhood to add into  $D$ . By the following proposition we ensure that Line 14 in Algorithm 2 always returns a valid vertex outside  $D^2$ .

---

<sup>2</sup> In some graphs there are vertices whose degree is 0. In these cases, we prevent such vertices from being removed.

**Proposition 8.** 1. If  $v$  is uncovered, then  $N[v] \cap D = \emptyset$ ;  
 2. For any uncovered vertex  $v$  s.t.  $d(v) \neq 0$ , there exists at least one vertex  $n \in N[v]$  s.t.  $free(n) = 1$ .

The proof follows the arguments in [4].

### 5 Speeding up of Best-Picking

In this section we present the details of our data structure that achieves the average complexity of  $ExchangeVertices(G, D)$  (Algorithm 2) of  $O(\bar{d}_2)$ .

**Score-Based Partition.** The idea is to partition the vertices wrt. their scores, *i.e.*, two vertices are in the same partition if they have the same score, otherwise they are in different partitions (see Fig. 1). Given a graph  $G = (V, E)$  and a candidate solution  $D$ , we implement the score-based partition on an array where each position holds a vertex. Besides, we maintain two types of pointers, *i.e.*,  $P_{gain}$  and  $P_{loss}$ , each of which points to the beginning of a specific partition.

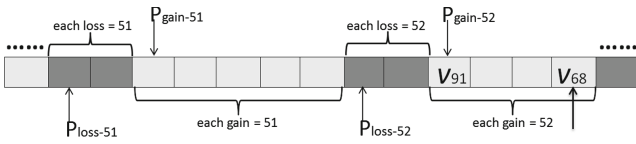


Fig. 1. Initial state before adding  $v_{68}$  into  $D$

**Algorithms and Implementations.** We use  $loss-k$  (resp.  $gain-k$ ) partition to denote the partition that contains vertices whose loss (resp. gain) is  $k$ . All the  $loss-k$  partitions are shown as dark regions, and all the  $gain-k$  partitions are shown as light ones. Then we use Algorithm 3 to find those vertices with the minimum loss.

---

**Algorithm 3:** randomMinLossVertex

---

**input** : A sequence of score-based partitions  
**output**: A random vertex  $v \in D$  with the minimum loss

```

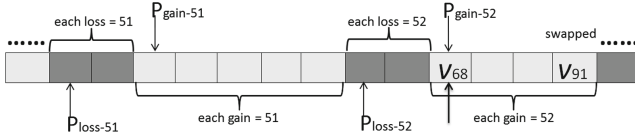
1  $k \leftarrow 0$ ;
2 while the  $loss-k$  partition is empty do  $k \leftarrow k + 1$ 
   return a random vertex in the  $loss-k$  partition;

```

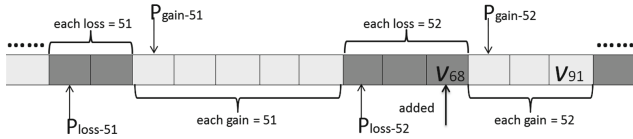
---

Since  $loss(v) \leq d(v) + 1$  for any  $v \in V$ , we have

**Proposition 9.** *If Algorithm 3 returns a vertex  $v$ , then the complexity of its execution is  $O(d(v))$ .*



**Fig. 2.** Swapping vertex  $v_{68}$  and  $v_{91}$



**Fig. 3.** Moving pointer  $P_{gain-52}$

At the beginning when  $D$  is empty, there are no dark regions in our data structure, so initializing the partitions is equivalent to sorting the vertices into a non-decreasing order wrt. their *gain*. Notice that at this time,  $gain(v) = d(v) + 1$  for all  $v \in V$ . By Proposition 4, we have

**Proposition 10.** *The complexity of initializing the partitions is  $O(|V|)$ .*

In both construction and local search phases, our algorithm will repeatedly add vertices into  $D$  or remove vertices from  $D$ , until some cutoff arrives. When a vertex  $v$  is added or removed, we have to maintain the *gain* or *loss* wrt. the vertices in  $N^2(v)$ . There are two cases in which a particular vertex, say  $v$ , has to be moved from one partition to another: (1) adding (resp. removing)  $v$  into (resp. from)  $D$ ; (2) increasing/decreasing  $gain(v)/loss(v)$ . Thus the core operation is to move a vertex  $v$  to an adjacent partition.

Now we show how to do this with an example (See Figs. 1, 2 and 3). In this example, we are to add  $v_{68}$  into  $D$ . Before  $v_{91}$  and  $v_{68}$  are in the *gain-52* partition and thus their *gain* is 52 (Fig. 1). Notice that after being added,  $v_{68}$ 's *loss* will become 52, *i.e.*, it should be in the *loss-52* partition. Thus the operation is like this: (1)  $v_{68}$  is swapped with  $v_{91}$  (Fig. 2); (2)  $P_{gain-52}$  is moved (Fig. 3).

We define `placeVertexIntoD( $v$ )` as the procedure that moves  $v$  from some *gain- $k$*  partition to the respective *loss- $k$*  partition. Also we define `dec_gain_move( $v$ )` to be the procedure that moves  $v$  from some *gain- $k$*  partition to the respective *gain- $(k-1)$*  partition. Likely we define `dec_loss_move( $v$ )`, `inc_gain_move( $v$ )`, `inc_loss_move( $v$ )` and `placeVertexOutFromD( $v$ )`. Then we have

**Proposition 11.** *All the procedures above are of  $O(1)$  complexity.*



**Complexity Analysis.** When a vertex  $v$  is added or removed, we simply need to maintain the *gain* or *loss* wrt. the vertices in  $N^2(v)$ . Hence we have

**Theorem 2.** *Suppose that each vertex has equal probability to be added or removed, then the average complexity of  $ExchangeVertices(G, D)$  (Algorithm 2) is  $O(\bar{d}_2)$ .*

**Table 1.** Results on large graphs where FastWMDS and ScBppw found different  $D_{min}$  or  $D_{avg}$

Graph	FastMWDS $D_{min}(D_{avg})$	ScBppw $D_{min}(D_{avg})$	$\delta_{min}$ ( $\delta_{avg}$ )
aff-orkut-user2groups	796325 (796405.9)	<b>791051 (791057.0)</b>	5274 (5348.9)
bn-human-BNU_1	<b>1189958 (1189968.3)</b>	1190244 (1190256.1)	-286 (-287.8)
ca-coauthors-dblp	<b>35626 (35634.3)</b>	37060 (37090.3)	-1434 (-1456.0)
ca-hollywood-2009	50752 (50928.2)	<b>50334 (50364.5)</b>	418 (563.7)
ca-MathSciNet	<b>65572 (65572.0)</b>	65594 (65598.0)	-22 (-26.0)
channel-500	403347 (405584.3)	<b>392298 (392409.6)</b>	11049 (13174.7)
dbpedia-link*	1537046 (1537073.6)	<b>1536656 (1536657.3)</b>	390 (416.3)
delaunay_n22	744846 (746514.5)	<b>689101 (689191.1)</b>	55745 (57323.4)
delaunay_n23	1514747 (1516652.2)	<b>1378214 (1378400.0)</b>	136533 (138252.2)
delaunay_n24*	3044234 (3045694.7)	<b>2756143 (2756488.0)</b>	288091 (289206.7)
friendster	656866 (656992.3)	<b>656464 (656466.4)</b>	402 (525.9)
hugebubbles-00020*	7464363 (7464956.0)	<b>6809676 (6810434.5)</b>	654687 (654521.5)
hugetrace-00010*	3849625 (3850125.7)	<b>3392608 (3393012.3)</b>	457017 (457113.4)
hugetrace-00020*	5111615 (5112821.7)	<b>4508857 (4509283.6)</b>	602758 (603538.1)
inf-europe_osm*	18854839 (18855830.6)	<b>17007576 (17009440.0)</b>	1847263 (1846390.6)
inf-germany_osm*	4236871 (4238019.7)	<b>3846526 (3846836.6)</b>	390345 (391183.1)
inf-roadNet-CA	628942 (632080.1)	<b>595745 (595800.1)</b>	33197 (36280.0)
inf-roadNet-PA	345733 (346350.5)	<b>332071 (332138.1)</b>	13662 (14212.4)
inf-road-usa*	8400680 (8401874.2)	<b>7852863 (7853454.2)</b>	547817 (548420.0)
rec-epinions	<b>9598 (9598.0)</b>	9599 (9599.0)	-1 (-1.0)
rgg-n.2.23_s0	739203 (741106.5)	<b>687540 (687945.5)</b>	51663 (53161.0)
rgg-n.2.24_s0	4102680 (4103927.2)	<b>4006264 (4006471.4)</b>	96416 (97455.8)
rt-retweet-crawl	75740 ( <b>75740.0</b> )	75740 (75740.1)	0 (-0.1)
sc-ldoor	<b>62473 (62484.7)</b>	64912 (66171.9)	-2439 (-3687.2)
sc-pwtk	<b>4194 (4197.8)</b>	5479 (5504.5)	-1285 (-1306.7)
sc-rel9	<b>119531 (120439.9)</b>	124304 (129925.1)	-4773 (-9485.2)
soc-delicious	<b>55722 (55722.0)</b>	55726 (55727.8)	-4 (-5.8)
soc-digg	66155 ( <b>66155.0</b> )	66155 (66156.6)	0 (-1.6)
soc-flickr	<b>98062 (98062.3)</b>	98063 (98064.9)	-1 (-2.6)
soc-flickr-und	295773 (295790.9)	<b>295702 (295705.1)</b>	71 (85.8)
soc-FourSquare	60982 (60985.7)	<b>60979 (60979.0)</b>	3 (6.7)
soc-groups	1072250 (1072306.7)	<b>1071123 (1071124.0)</b>	1127 (1182.7)
soc-ljournal-2008	1015711 (1015933.3)	<b>1005983 (1005988.2)</b>	9728 (9945.1)

(continued)

**Table 1.** (continued)

Graph	FastMWDS $D_{min}(D_{avg})$	ScBppw $D_{min}(D_{avg})$	$\delta_{min}$ ( $\delta_{avg}$ )
soc-orkut-dir	98716 (99715.8)	<b>94012 (94037.5)</b>	4704 (5678.3)
soc-pokec	213149 (213241.0)	<b>207383 (207389.7)</b>	5766 (5851.3)
soc-youtube-snap	213122 (213131.0)	213122 ( <b>213123.3</b> )	0 (7.7)
socfb-A-anon	201691 (201699.5)	<b>201690 (201690.5)</b>	1 (9.0)
socfb-B-anon	187030 (187030.2)	187030 ( <b>187030.1</b> )	0 (0.1)
tech-as-skitter	182427 (182736.4)	<b>181852 (181869.8)</b>	575 (866.6)
tech-ip	160 (161.2)	<b>156 (157.1)</b>	4 (4.1)
tech-RL-caida	<b>40095 (40095.8)</b>	40142 (40152.9)	-47 (-57.1)
web-it-2004	<b>32997 (32997.0)</b>	32998 (32999.3)	-1 (-2.3)
web-wikipedia2009	348003 (348024.5)	<b>346676 (346682.4)</b>	1327 (1342.1)
web-wikipedia-growth	117626 (117663.0)	<b>116817 (116818.5)</b>	809 (844.5)
web-wikipedia_link_it	618963 (619083.8)	<b>617660 (617661.3)</b>	1303 (1422.5)
wikipedia_link_en*	23995928 (23995933.6)	<b>23995924 (23995924.0)</b>	4 (9.6)

## 6 Experimental Evaluations

We compared the overall performances of our solver ScBppw to FastWMDS on the benchmark instances in [24] and [16] which contain more than  $5 \times 10^5$  vertices. Notice that large graphs are the main challenge of current solvers, and their size will keep growing due to the amount of data available. There are some other MinDS solvers like SAMDS [14], CC<sup>2</sup>FS [25] and RLS-DS [7], but they do not materially change our conclusions below, because our preliminary experiments show that FastWMDS significantly outperforms them. Also we conducted experiments to evaluate the individual impacts.

### 6.1 Experimental Setup

FastWMDS was compiled by g++ 5.4.0 with O3 option while ScBppw was compiled by g++ 4.7.3 with O3 option. The experiments were conducted on a cluster equipped with Intel Xeon E5-2670 v3 2.3 GHz with 64 GB RAM, running CentOS6. Each solver was executed on each instance with seeds from 1 to 10. The cutoff was set to 1,000s. For each algorithm on each instance, we report the minimum size (“ $D_{min}$ ”) and averaged size (“ $D_{avg}$ ”) of the dominating sets found by the algorithm over the 10 runs. To make the comparisons clearer, we also report the difference (“ $\delta_{min}$ ”) between the minimum size returned by ScBppw and that returned by other solvers. Similarly  $\delta_{avg}$  represents the difference between the averaged sizes. A positive difference indicates that ScBppw performs better, while a negative value indicates the opposite. For the sake of space, we do not report results on those graphs where all solvers found the same  $D_{min}$  and  $D_{avg}$ . The best  $D_{min}$  and  $D_{avg}$  values are shown in **bold font**. For the sake of space, in each table we will write *bn-human-BNU\_1* in short

for *bn-human-BNU\_1-0025865\_session\_1-bg*, *channel-500* in short for *channel-500x100x100-b050*, and *soc-groups* in short for *soc-livejournal-user-groups*.

## 6.2 Main Results

Table 1 compares the overall performances of FastWMDS and ScBppw. From this table, we observe that:

1. ScBppw outperforms FastWMDS on most instances in terms of  $D_{min}$  and  $D_{avg}$ .
2. Further observations show that there are 11 graphs which contain more than  $10^7$  vertices. Among these graphs, ScBppw outperforms FastWMDS on 9 graphs (marked with \* in the table) in terms of both  $D_{min}$  and  $D_{avg}$ . On the other 2 graphs, *i.e.*, `soc-sinaweibo` and `socfb-uci-uni`, both solvers found the same  $D_{min}$  and  $D_{avg}$ . So ScBppw is more scalable than FastWMDS.

We also did similar experiments on those graphs in [24] and [16] whose vertex number lies between  $2 \times 10^4$  and  $5 \times 10^5$ . Experimental results show that ScBppw is comparable and complementary with FastWMDS. More specifically, they have different performances, and the number of graphs on which ScBppw performs better is roughly equal to the number of graphs on which FastWMDS performs better. In addition, both solvers outperforms the other MinDS solvers.

Notice that FastWMDS exploits a list of reduction rules to simplify the input graphs before doing local search, while our solver only adopts a simple greedy heuristic. Therefore we can conclude that pure local search is also competitive.

## 6.3 Individual Impacts

In what follows, we first modify our algorithm and develop several variants. Then we redo the experiments above and compare them in terms of average solution quality.

**Effectiveness of Score Checking.** We replace the SC strategy in ScBppw with the NVCC,  $CC^2$  and  $CC^2V3$  strategies, and develop three variants named NVCCBppw,  $CC^2$ Bppw and  $CC^2V3$ Bppw respectively.

When comparing NVCCBppw with ScBppw, we find that

1. ScBppw performs better on 19 instances;
2. NVCCBppw performs better on 14 instances.

This means that the SC strategy is more suitable than the NVCC strategy in our solver. The detailed results are shown in Table 2.

Furthermore, when comparing  $CC^2$ Bppw,  $CC^2V3$ Bppw with ScBppw, we find that

1.  $CC^2$ Bppw outperforms the other solvers on 0 instances;
2.  $CC^2V3$ Bppw outperforms the others on 6 instances;

**Table 2.** Results on large graphs where NVCCBppw and ScBppw found different  $D_{min}$  or  $D_{avg}$ 

Graph	NVCCBppw $D_{min}(D_{avg})$	ScBppw $D_{min}(D_{avg})$	$\delta_{min}$ ( $\delta_{avg}$ )
aff-orkut-user2groups	791055 (791063.7)	<b>791051 (791057.0)</b>	4 (6.7)
ca-coauthors-dblp	37069 (37095.3)	<b>37060 (37090.3)</b>	9 (5.0)
ca-hollywood-2009	50418 (50445.9)	<b>50334 (50364.5)</b>	84 (81.4)
delaunay_n22	<b>689100 (689190.7)</b>	689101 (689191.1)	-1 (-0.4)
delaunay_n23	<b>1378207 (1378398.3)</b>	1378214 (1378400.0)	-7 (-1.7)
delaunay_n24	<b>2756090 (2756470.0)</b>	2756143 (2756488.0)	-53 (-18.0)
hugebubbles-00020	6809994 (6810453.1)	<b>6809676 (6810434.5)</b>	318 (18.6)
hugetrace-00010	3392673 (3393040.0)	<b>3392608 (3393012.3)</b>	65 (27.7)
hugetrace-00020	4508901 (4509384.1)	<b>4508857 (4509283.6)</b>	44 (100.5)
inf-europe.osm	<b>17006505 (17009166.9)</b>	17007576 (17009440.0)	-1071 (-273.1)
inf-germany.osm	<b>3846466 (3846831.3)</b>	3846526 (3846836.6)	-60 (-5.3)
inf-roadNet-CA	595868 (595938.4)	<b>595745 (595800.1)</b>	123 (138.3)
inf-roadNet-PA	332093 (332190.3)	<b>332071 (332138.1)</b>	22 (52.2)
inf-road-usa	7855630 (7856923.6)	<b>7852863 (7853454.2)</b>	2767 (3469.4)
rgg_n_2_23_s0	<b>687538 (687944.9)</b>	687540 (687945.5)	-2 (-0.6)
rgg_n_2_24_s0	4006264 ( <b>4006469.2</b> )	4006264 (4006471.4)	0 (-2.2)
sc-ldoor	<b>64485 (65997.7)</b>	64912 (66171.9)	-427 (-174.2)
sc-rel9	<b>124210 (129763.8)</b>	124304 (129925.1)	-94 (-161.3)
soc-delicious	55726 ( <b>55727.4</b> )	55726 (55727.8)	0 (-0.4)
soc-digg	66155 ( <b>66156.5</b> )	66155 (66156.6)	0 (-0.1)
soc-flickr	98064 (98065.7)	<b>98063 (98064.9)</b>	1 (0.8)
soc-FourSquare	60979 (60979.4)	60979 ( <b>60979.0</b> )	0 (0.4)
soc-orkut-dir	<b>93999 (94032.2)</b>	94012 (94037.5)	-13 (-5.3)
soc-pokec	207559 (207597.9)	<b>207383 (207389.7)</b>	176 (208.2)
soc-youtube-snap	213122 (213123.5)	213122 ( <b>213123.3</b> )	0 (0.2)
socfb-A-anon	201690 (201690.6)	201690 ( <b>201690.5</b> )	0 (0.1)
socfb-B-anon	187030 (187030.3)	187030 ( <b>187030.1</b> )	0 (0.2)
tech-as-skitter	181874 (181890.1)	<b>181852 (181869.8)</b>	22 (20.3)
tech-ip	<b>155 (156.7)</b>	156 (157.1)	-1 (-0.4)
tech-RL-caida	40143 ( <b>40152.5</b> )	<b>40142 (40152.9)</b>	1 (-0.4)
web-wikipedia2009	<b>346675 (346687.9)</b>	346676 ( <b>346682.4</b> )	-1 (5.5)
web-wikipedia-growth	116817 (116818.8)	116817 ( <b>116818.5</b> )	0 (0.3)

3. ScBppw outperforms the others on 32 instances.

This indicates that the SC strategy is more effective than the  $CC^2$  and  $CC^2V3$  strategies. For the sake of space, we omit the detailed results here. Overall, the SC strategy plays an essential role in our solver.

Now we analyze the performances. (1) Apart from the tabu strategy, we find that there are two powerful diversification strategies in FastWMDS: a frequency based scoring function and a best-from-multiple-selection heuristic with some random walks. In contrast, besides the tabu strategy, ScBppw only exploits two weak diversification strategies: (i) best-picking with probabilistic walk; (ii)

randomly selecting an uncovered vertex. Since (i) and (ii) are weak diversification strategies, an effective tabu strategy in our solver should be stronger than the  $CC^2V3$  strategy in FastWMDS. Indeed, NVCC and SC greatly outperform  $CC^2$  and  $CC^2V3$  in our solver. (2) When a vertex  $v$  is added or removed, SC can release some vertices whose distance from  $v$  is 2. However, NVCC tends to release more vertices whose distance from  $v$  is 1. Thus SC is more likely to search in a bigger area while NVCC is more cautious in current areas. Therefore their performances are quite different.

**Effectiveness of Best-Picking with Probabilistic Walk.** We modify this strategy and develop two variants, ScBprw (Score Checking and Best-picking with Random Walk) and ScBppw-nlsp (ScBppw with no *lastStepImproved*) as follows. To develop ScBprw, we replace the probability selection in ScBppw with random walk, which means every vertex in  $D$  has equal probability to be removed. On the other hand, to develop ScBppw-nlsp, we remove the *lastStepImproved* predicate from ScBppw so that our algorithm will mark and check the hash table regardless of the *lastStepImproved* value. More specifically, we remove lines 1, 2, 8, 9, and 17 to 21, so as to develop ScBppw-nlsp. Then we compare these two variants with ScBppw. Experimental results show that

1. ScBprw outperforms the other solvers on 7 instances;
2. ScBppw-nlsp outperforms the others on 7 instances;
3. ScBppw outperforms the others on 21 instances.

Moreover, we remove the probabilistic selection component and develop a variant called ScDS, which will always do best-picking in the removing stage. Experimental results show ScBppw significantly outperforms ScDS as well.

Overall, we conclude that the probabilistic selection as well as the *lastStepImproved* predicate play an important role in ScBppw. For the sake of space, we omit the detailed results here.

## 7 Conclusions

In this paper, we have developed a local search MinDS solver named ScBppw. Experimental results show that our solver outperforms state-of-the-art on large graphs. In particular, it makes a substantial improvement on those graphs that contain more than  $10^7$  vertices. The main contributions include: (1) a tabu strategy based on score change; (2) a best-picking with probabilistic walk strategy; (3) a data structure to accelerate best-picking.

For future works, we will develop efficient heuristics to solve the MinDS problem on massive graphs which are too large to be stored in the main memory and has to be accessed through disk IOs.

## References

1. Abseher, M., Dusberger, F., Musliu, N., Woltran, S.: Improving the efficiency of dynamic programming on tree decompositions via machine learning. In: IJCAI 2015, pp. 275–282 (2015)
2. Alber, J., Fellows, M.R., Niedermeier, R.: Efficient data reduction for DOMINATING SET: a linear problem kernel for the planar case. In: Penttonen, M., Schmidt, E.M. (eds.) SWAT 2002. LNCS, vol. 2368, pp. 150–159. Springer, Heidelberg (2002). [https://doi.org/10.1007/3-540-45471-3\\_16](https://doi.org/10.1007/3-540-45471-3_16)
3. Barabasi, A.L., Albert, R.: Emergence of scaling in random networks. *Science* **286**(5439), 509–512 (1999)
4. Cai, S., Su, K., Luo, C., Sattar, A.: NuMVC: an efficient local search algorithm for minimum vertex cover. *J. Artif. Intell. Res. (JAIR)* **46**, 687–716 (2013)
5. Cai, S., Su, K., Sattar, A.: Local search with edge weighting and configuration checking heuristics for minimum vertex cover. *Artif. Intell.* **175**(9–10), 1672–1696 (2011)
6. Campan, A., Truta, T.M., Beckerich, M.: Fast dominating set algorithms for social networks. In: Proceedings of the 26th Modern AI and Cognitive Science Conference 2015, pp. 55–62 (2015)
7. Chalupa, D.: An order-based algorithm for minimum dominating set with application in graph mining. *Inf. Sci.* **426**, 101–116 (2018)
8. Chung, F.R., Lu, L.: *Complex graphs and networks*, vol. 107 (2006)
9. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: *Introduction to Algorithms*, 3rd edn. MIT Press, Cambridge (2009)
10. Eubank, S., Kumar, V.S.A., Marathe, M.V., Srinivasan, A., Wang, N.: Structural and algorithmic aspects of massive social networks. In: Proceedings of SODA 2004, pp. 718–727 (2004)
11. Fan, Y., Li, C., Ma, Z., Wen, L., Sattar, A., Su, K.: Local search for maximum vertex weight clique on large sparse graphs with efficient data structures. In: Kang, B.H., Bai, Q. (eds.) AI 2016. LNCS (LNAI), vol. 9992, pp. 255–267. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-50127-7\\_21](https://doi.org/10.1007/978-3-319-50127-7_21)
12. Fan, Y., Li, N., Li, C., Ma, Z., Latecki, L.J., Su, K.: Restart and random walk in local search for maximum vertex weight cliques with evaluations in clustering aggregation. In: IJCAI 2017, Melbourne, Australia, 19–25 August 2017, pp. 622–630 (2017)
13. Haynes, T.W., Hedetniemi, S.M., Hedetniemi, S.T., Henning, M.A.: Domination in graphs applied to electric power networks. *SIAM J. Discrete Math.* **15**(4), 519–529 (2002)
14. Hedar, A.R., Ismail, R.: Simulated annealing with stochastic local search for minimum dominating set problem. *Int. J. Mach. Learn. Cybern.* **3**(2), 97–109 (2012)
15. Hoos, H.H., Stützle, T.: Stochastic local search. In: *Handbook of Approximation Algorithms and Metaheuristics* (2007)
16. Jiang, H., Li, C., Manyà, F.: An exact algorithm for the maximum weight clique problem in large graphs. In: AAAI, California, USA, pp. 830–838 (2017)
17. Li, C.M., Huang, W.Q.: Diversification and determinism in local search for satisfiability. In: Bacchus, F., Walsh, T. (eds.) SAT 2005. LNCS, vol. 3569, pp. 158–172. Springer, Heidelberg (2005). [https://doi.org/10.1007/11499107\\_12](https://doi.org/10.1007/11499107_12)
18. Ma, Z., Fan, Y., Su, K., Li, C., Sattar, A.: Local search with noisy strategy for minimum vertex cover in massive graphs. In: Booth, R., Zhang, M.-L. (eds.) PRICAI 2016. LNCS (LNAI), vol. 9810, pp. 283–294. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-42911-3\\_24](https://doi.org/10.1007/978-3-319-42911-3_24)

19. Murray, R.S.: *Theory and Problems of Probability and Statistics*. McGraw-Hill Book and Co., Singapore (1975)
20. Nacher, J.C., Akutsu, T.: Minimum dominating set-based methods for analyzing biological networks. *Methods* **102**, 57–63 (2016)
21. Potluri, A., Bhagyati, C.: Novel morphological algorithms for dominating sets on graphs with applications to image analysis. In: Barneva, R.P., Brimkov, V.E., Aggarwal, J.K. (eds.) *IWCIA 2012*. LNCS, vol. 7655, pp. 249–262. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-34732-0\\_19](https://doi.org/10.1007/978-3-642-34732-0_19)
22. Samuel, H., Zhuang, W., Preiss, B.: DTN based dominating set routing for manet in heterogeneous wireless networking. *Mob. Netw. Appl.* **14**(2), 154–164 (2009)
23. Shen, C., Li, T.: Multi-document summarization via the minimum dominating set. In: *COLING 2010*, Beijing, China, pp. 984–992 (2010)
24. Wang, Y., Cai, S., Chen, J., Yin, M.: A fast local search algorithm for minimum weight dominating set problem on massive graphs. In: *IJCAI 2018*, pp. 1514–1522 (2018)
25. Wang, Y., Cai, S., Yin, M.: Local search for minimum weight dominating set with two-level configuration checking and frequency based scoring function. *J. Artif. Intell. Res.* **58**, 267–295 (2017)
26. Yao, B., Fei-Fei, L.: Action recognition with exemplar based 2.5D graph matching. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) *ECCV 2012*. LNCS, vol. 7575, pp. 173–186. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-33765-9\\_13](https://doi.org/10.1007/978-3-642-33765-9_13)



# Multi-level Graph Compression for Fast Reachability Detection

Shikha Anirban<sup>(✉)</sup>, Junhu Wang, and Md. Saiful Islam

Griffith University, Gold Coast, Australia  
shikha.anirban@griffithuni.edu.au

**Abstract.** Fast reachability detection is one of the key problems in graph applications. Most of the existing works focus on creating an index and answering reachability based on that index. For these approaches, the index construction time and index size can become a concern for large graphs. More recently query-preserving graph compression has been proposed and searching reachability over the compressed graph has been shown to be able to significantly improve query performance as well as reducing the index size. In this paper, we introduce a multilevel compression scheme for DAGs, which builds on existing compression schemes, but can further reduce the graph size for many real-world graphs. We propose an algorithm to answer reachability queries using the compressed graph. Extensive experiments with two existing state-of-the-art reachability algorithms and 10 real-world datasets demonstrate that our approach outperforms the existing methods.

**Keywords:** Modular decomposition · Graph compression · Reachability queries · Algorithms

## 1 Introduction

The reachability query, which asks whether there exists a path from one vertex to another in a directed graph, finds numerous applications in graph and network analysis. Such queries can be answered by graph traversal using either breadth-first or depth-first search in time  $O(|E| + |V|)$  without pre-processing (where  $V$  and  $E$  are the vertex set and edge set respectively), or in constant time if we pre-compute and store the transitive closure of each vertex, which takes  $O(|V||E|)$  time and  $O(|V|^2)$  space. Unfortunately, neither of these approaches is feasible for applications that need to process large graphs with limited memory. Over the last decades, the problem has been extensively studied and many advanced algorithms have been proposed, with most of them relying on building smart indexes that can strike a balance between online query processing time and offline index construction time (and index size).

More recently, researchers recognized that it is possible to reduce the graph size by graph compression without losing reachability information, and the compressed graph can help speedup query processing as well as reduce index size



and index construction time. Specially, Fan *et al.* [6] define equivalence classes of vertices with respect to reachability queries, and compress a graph by merging all vertices in an equivalence class into a single vertex. However, finding all equivalence classes is very time-consuming. Zhou *et al.* [21] propose an efficient algorithm to do a *transitive reduction* which turns a directed acyclic graph (DAG) into a DAG without redundant edges, after that the *equivalence reduction* of [6] can be done much more efficiently. The resulting graph  $G^\epsilon$  after transitive reduction and equivalence reduction over the original graph  $G$  can be a much smaller graph that retains all reachability information, and it was experimentally verified that for many real-world graphs, searching for reachability over  $G^\epsilon$  can be much faster than searching over  $G$  using state-of-the-art algorithms.

This paper builds on the work of [21]. We observe that after the removal of redundant edges many *linear chains* will be generated. Based on this, we propose a multi-level reachability - preserving compression method that can further reduce the size of the graph obtained by the method in [21]. Our compression utilizes a slightly modified concept of *module* [13], and constructs a modular decomposition tree. We show how to use the decomposition tree to answer reachability queries over the original graph efficiently. Furthermore, the decomposition tree usually takes very small space. We make the following contributions:

1. We define a new concept of *module*, based on which we propose a multilevel graph compression scheme that compresses graphs into a smaller graph  $G_c$ .
2. We organize the modules into a hierarchical structure called *modular decomposition tree*, and propose an efficient algorithm to utilize the tree to answer reachability queries.
3. We conduct extensive experiments with real-world graphs that demonstrate the advantages of our proposed approach.

The remainder of this paper is organized as follows. We first discuss related works in Sect. 2 and present the preliminaries in Sect. 3. Then we give an overview of our approach and provide the theoretical foundations in Sect. 4, followed by the detailed algorithms in Sect. 5. Our experimental results are given in Sect. 6. We conclude our paper in Sect. 7.

## 2 Related Work

As briefly mentioned in Sect. 1, existing approaches for answering reachability queries can be classified into index-based and compression-based.

**Index-Based Approach:** The index-based algorithms create labels for the vertices, such labels contain the reachability information. These algorithms can be divided into *Label-only* and *Label+G* methods [18]. The label-only [1, 3, 4, 7, 9–12, 19] methods use the labels of source and destination vertices only to answer reachability. Agrawal [1] proposed tree cover approach that creates an optimal

spanning tree to create index. Here, an interval for each vertex is created. A reachability query is answered as true if the interval of target is contained in the interval of source vertex. The index construction time and index size both are high in this approach. A *chain cover* approach is first proposed in [7] where the entire graph is divided into a number of pairwise disjoint chains to create the index. The label of each vertex contains a minimal successor list containing their chain number and position in the chain. A vertex  $u$  will be reachable to  $v$  if label of  $u$  contains a pair  $(k, j)$  and  $v$  has an index pair  $(i, j)$  such that  $i \geq k$ . This chain cover approach is later improved in [3]. Path tree [9] uses the similar concept of chain cover that uses paths to create index and has smaller index size than chain cover. The recent approaches DL [10], PLL [19] and TF [4] use the concept of 2-hop labeling proposed in [5]. In 2-hop labeling, a label is created for each vertex containing the subset of vertices that it can reach ( $L_{out}$ ) as well as the subset of vertices that can reach it ( $L_{in}$ ). Vertex  $u$  can reach vertex  $v$  if  $L_{out}(u) \cap L_{in}(v) \neq \emptyset$ . [11] uses the concept of chain cover to improve 2-hop and proposes a 3-hop labeling that creates a transitive closure contour ( $Con(G)$ ) of graph  $G$  using chain decomposition, and then applies 2-hop techniques. Path-hop [2] improves 3-hop by replacing the chain decomposition with a spanning tree. TF [4] proposes a topological folding approach for 2-hop labeling that can significantly reduce the index size as well as the query time.

The Label+G approaches include [14–18, 20] which require online searching of data graph  $G$  if the query can not be answered from labels. [16] uses interval labeling over a spanning tree and performs DFS online if needed. Grail [20] and Ferrari [14] use multiple interval instead of single interval label for each vertex over the spanning tree. Feline [17] creates coordinates  $i(u) = (X_u, Y_u)$  for a vertex  $u$  and answers reachability from  $u$  to  $v$  as true if the area of  $i(v)$  is contained in that of  $i(u)$ . Feline also uses interval labeling over spanning tree and compares topological levels of  $u$  and  $v$  as additional pruning strategy to reduce DFS search. IP<sup>+</sup> [18] uses independent permutation numbering to label each vertex. Feline and IP<sup>+</sup> show significant improvement on query time and require less index construction time and smaller index size. BFL [15] proposes a Bloom-Filter Labeling to further improve the performance of IP<sup>+</sup>.

**Compression-Based Approach:** Graph compression based works include Scarab [8], Equivalence reduction [6] and DAG reduction [21]. Scarab [8] compresses the original graph by creating a reachability backbone that carries the major reachability information. To find reachability from vertex  $u$  to vertex  $v$  the algorithm needs access to a list of local outgoing backbone vertices of  $u$  and local incoming backbone vertices of  $v$ . The algorithm then performs a forward BFS for  $u$  and backward BFS for  $v$  on the original graph to answer reachability from  $u$  to  $v$ . If the answer is false then it checks whether any outgoing backbone vertex of  $u$  can reach any incoming backbone vertex of  $v$  in the reachability backbone, if yes, then  $u$  can reach  $v$ . Scarab requires large index size with high time complexity. Equivalence reduction [6] reduces the graph by merging

equivalent vertices into a single vertex. Two vertices are equivalent if they have the same ancestors and same descendants. The algorithm requires high equivalence class construction time. DAG reduction [21] improves the construction time of equivalence classes by doing a transitive reduction of graph first.

Our work is different from the previous works in that we not only consider equivalent classes, but also linear chains, when compressing the graph, and to the best of our knowledge, none of the previous works uses multi-level compression and modular decomposition tree in reachability queries.

### 3 Preliminaries

We consider directed graphs in this paper. For any directed graph  $G$ , we will use  $V_G$  and  $E_G$  to denote the vertex set and the edge set of  $G$ , respectively. Given two vertices  $u$  and  $v$  in  $G$ , if there is a path from  $u$  to  $v$ , we say  $v$  is *reachable* from  $u$ , or equivalently,  $u$  can *reach*  $v$ . We use  $u \rightsquigarrow_G v$  to denote  $u$  can reach  $v$  in graph  $G$ . Given directed graph  $G$  and vertices  $u$  and  $v$  in  $G$ , a reachability query from  $u$  to  $v$ , denoted  $?u \rightsquigarrow_G v$ , asks whether  $v$  is reachable from  $u$  in  $G$ .

A directed acyclic graph (DAG) is a directed graph without cycles. In the literature, most works on reachability queries assume the graph  $G$  is a DAG, because if it is not, it can be converted into a DAG by merging all vertices in a strongly connected component into a single vertex, and vertices in a strongly connected component can all reach each other. In this work, we also assume the graph  $G$  is a DAG.

If  $(u, v)$  is an edge in DAG  $G$ , we say  $u$  is a *parent* of  $v$ , and  $v$  is a *child* of  $u$ . For any vertex  $u \in V_G$ , we will use  $parent(u, G)$  and  $child(u, G)$ , respectively, to denote the set of parents of  $u$  and the set of children of  $u$  in  $G$ . We will also use  $anc(u, G)$  and  $des(u, G)$  to denote the set of ancestors of  $u$  and the set of descendants of  $u$  in  $G$ , respectively. When  $G$  is clear from the context, we will use the abbreviations  $parent(u)$ ,  $child(u)$ ,  $anc(u)$ , and  $des(u)$  for  $parent(u, G)$ ,  $child(u, G)$ ,  $anc(u, G)$ , and  $des(u, G)$  respectively.

Let  $M$  be a subset of vertices in  $G$ . For any vertex  $u \in M$  and a parent vertex  $u'$  of  $u$ , we say  $u'$  is an *external* parent of  $u$  (with respect to  $M$ ) if  $u' \in parent(u) - M$ . Similarly, we define an *external* child (resp. ancestor, descendent) of  $u$  with respect to  $M$  as a vertex in  $child(u) - M$  (resp.  $anc(u) - M$ ,  $des(u) - M$ ).

**Redundant Edges.** Suppose  $(u, v)$  is an edge in  $G$ . If there is a path of length greater than 1 from  $u$  to  $v$ , then  $(u, v)$  is *redundant* for reachability queries, that is, removing  $(u, v)$  from  $G$  will not affect the answer to any reachability queries.

The redundant edges can be efficiently identified and removed by a *transitive reduction* algorithm proposed in [21]. The following lemma is shown in [21]:

**Lemma 1.** *Suppose  $G$  is a DAG without redundant edges, then for any two vertices  $u$  and  $v$  in  $G$ ,  $parent(u) = parent(v)$  if and only if  $anc(u) = anc(v)$ ;  $child(u) = child(v)$  if and only if  $des(u) = des(v)$ .*

**Equivalence Class.** Two vertices  $u$  and  $v$  are said to be *equivalent* if they have the same ancestors and the same descendants, that is,  $anc(u) = anc(v)$ ,  $des(u) = des(v)$  [6]. Because of Lemma 1, if  $G$  does not have redundant edges, then  $u$  and  $v$  are equivalent if and only if they have the same parents and same children. The equivalent vertices form an equivalence class. It is easy to see that all vertices in the same equivalence class have the same reachability properties, that is, if  $u$  is in an equivalence class, then for any other vertex  $u'$ ,  $u$  can reach  $u'$  (resp.  $u$  is reachable from  $u'$ ) if and only if every vertex  $v$  in the same equivalence class can reach  $u'$  (resp. is reachable from  $u'$ ).

Also as observed in [21], if  $G$  has no redundant edges, then all vertices in an equivalence class form an independent set, that is, there are no edges between the vertices in the same equivalence class.

**Lemma 2.** *Suppose  $G$  is a DAG without redundant edges, then every equivalent class is an independent set.*

**Modular Decomposition.** The modular decomposition [13] of a directed graph  $G$  partitions the vertex set into a hierarchy of *modules*, where a module is conventionally defined as follows.

**Definition 1.** *Let  $M$  be a set of vertices in  $G$ . We call  $M$  a module of  $G$  if all vertices in  $M$  share the same external parents and the same external children. In other words, for any  $u, v \in M$ ,  $parent(u) - M = parent(v) - M$  and  $child(u) - M = child(v) - M$ .*

It is easy to see that a singleton set is a module and the set of all vertices in  $G$  is also a module. These modules are called *trivial* modules. Let  $G$  be a DAG that has no redundant edges. By Lemma 1, an equivalent class is also a module, and by Lemma 2, such a module is an independent set. In the literature, modules that are independent sets are referred to as *parallel* modules.

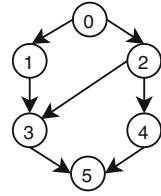
## 4 Overview of Our Approach

The basic idea of our method is to compress the graph without losing reachability information. We use *modular decomposition*, however the definition of modules has been slightly modified from that found in the literature, in order to help with reachability queries.

**Definition 2.** *A module in a DAG  $G$  is a set of vertices  $M \subseteq V_G$  that have identical external ancestors and identical external descendants. In other words, for any two vertices  $u, v \in M$ ,  $anc(u) - M = anc(v) - M$ , and  $des(u) - M = des(v) - M$ .*

Our module is an extension of the conventional module defined in Definition 1, that is, conventional modules are also modules by our definition. This is because vertices that share the same external parents also share the same external ancestors, and vertices that share the same external children also share the same external descendants. However, the converse is not true. For example, in the graph shown in Fig. 1, the set of vertices  $\{1, 2, 3, 4\}$  is a module by our definition. However, it is not a module according to the conventional definition. In what follows, when we say a module, we mean a module by Definition 2, unless explicitly stated otherwise.

In this work, we are interested in two special types of modules, referred to as *parallel modules* and *linear modules* respectively. A parallel module is a module that is an independent set, and a linear module is one that consists of a *chain* of vertices  $v_1, \dots, v_k$  such that there is an edge  $(v_i, v_{i+1})$  for all  $i \in [1, k - 1]$ . These modules have the following properties.



**Fig. 1.** Example DAG, where the vertices 1, 2, 3, 4 have same external ancestors and same external descendants, but not the same external parents and same external children

**Lemma 3.** *Suppose  $G$  is a DAG that does not have redundant edges. (1) If  $M$  is a parallel module of  $G$ , then all vertices in  $M$  have the same parents and same children. (2) If  $M$  is a linear module consisting of the chain  $v_1, \dots, v_k$ , then for each  $i \in [2, k]$ ,  $v_{i-1}$  is the only parent of  $v_i$ , and  $v_i$  is the only child of  $v_{i-1}$ .*

*Proof.* (1) Let  $M$  be a parallel module. By definition  $M$  is an independent set, and all the vertices have the same external ancestors and the same external descendants. Since  $M$  is an independent set, it is impossible for any vertex in  $M$  to have an ancestor or descendent in  $M$ , therefore, all the vertices have the same ancestors and the same descendants (both external and internal). By Lemma 1, all vertices in  $M$  have the same parents and the same children.

(2) Let  $M$  be a linear module consisting of the chain  $v_1, \dots, v_k$ . For any  $i \in [2, k]$ , if  $v_i$  has a parent  $u$  that is not  $v_{i-1}$ , then there are two possible cases. The first case is that  $u$  is also in  $M$ , that is  $u$  is one of  $v_{i+1}, \dots, v_k$ . This contradict the assumption that  $G$  is a DAG since there will be a cycle. The second case is that  $u$  is not in  $M$ . In this case, by the definition of a module,  $u$  must be an ancestor of  $v_1$ , that is, there will be a path from  $u$  to  $v_i$  with length at least 2. Hence the edge  $(u, v_i)$  would be redundant, contradicting the assumption that there are no redundant edges in  $G$ . This proves  $v_{i-1}$  is the only parent of  $v_i$ . Similarly we can prove  $v_i$  is the only child of  $v_{i-1}$ .

In Fig. 2(a), the vertices  $v_1, v_2, v_3$  form a parallel module. In Fig. 2(b), the vertices  $v_1, v_2, v_3$  form a linear module. Note, however, the set  $\{v_4, v_1, v_2, v_3, v_6\}$  in Fig. 2(b) is not a linear module.

It is worth noting that each single vertex forms a parallel module as well as a linear module. These modules are referred to as *trivial* modules, along with the module that consists of all of the vertices in  $G$ . According to Lemma 3, a parallel module is an equivalence class, if  $G$  is a DAG that has no redundant edges.

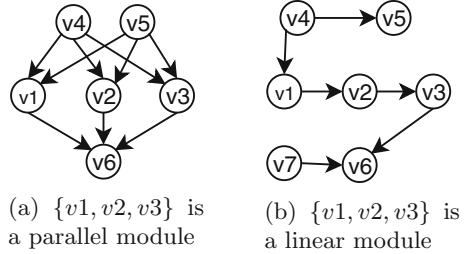


Fig. 2. Example of modules

Note that if two vertices are in the same linear module, then their reachability depends on their relative positions in the chain. If they are in the same parallel module, then they cannot reach each other, as shown in the lemma below.

**Lemma 4.** *Let  $G$  be a DAG without redundant edges, and  $u, v$  be vertices in the same parallel module of  $G$ , then  $u$  cannot reach  $v$  in  $G$ .*

*Proof.* Let the parallel module that contains  $u$  and  $v$  be  $M$ . If the lemma is not true, there will be a path  $u, v_1, \dots, v_s, v$  from  $u$  to  $v$ . Since  $M$  is an independent set,  $v_1$  and  $v_s$  cannot be in  $M$ . Hence  $v_1$  is an external child of  $u$ , and  $v_s$  is an external parent of  $v$ . By Lemma 3 and the definition of modules,  $v_1$  must be a child of  $v$  and  $v_s$  must be a parent of  $u$ . Therefore there will be a cycle, contradicting the assumption that  $G$  is a DAG. Hence the proof.

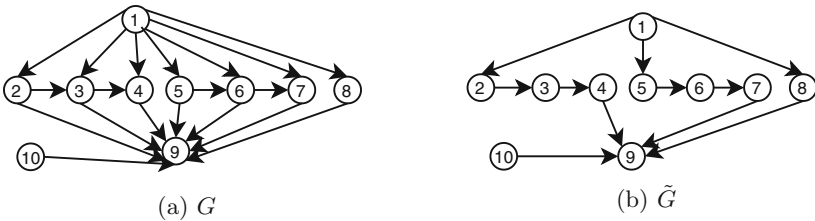


Fig. 3. (a) A DAG  $G$  and (b) the DAG  $\tilde{G}$  after transitive reduction

A set of vertices may be in multiple parallel (or linear) modules, e.g., in the graph shown in Fig. 2(a),  $\{v1, v2\}$  and  $\{v1, v2, v3\}$  are both parallel modules. However, we are only interested in the *maximal* modules as defined below.

**Definition 3.** *A parallel (resp. linear) module  $M$  is said to be maximal if there is no other parallel (resp. linear) module  $M'$  such that  $M \subset M'$ .*

For example,  $\{v1, v2, v3\}$  is a maximal parallel module in Fig. 2(a), and it is a maximal linear module in Fig. 2(b).

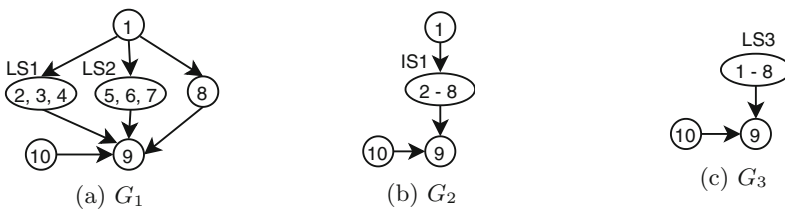
Note that two different maximal parallel modules of  $G$  cannot have overlaps, and two different maximal linear modules cannot have overlaps. Furthermore, there cannot exist a non-trivial parallel module and a non-trivial linear module

such that they have a common vertex. In other words, each vertex can belong to at most one non-trivial parallel or linear module.

**Multi-level Compression and Modular Decomposition Tree.** To utilize parallel and linear modules in reachability search, we perform a *multi-level compression* of the original graph  $G$ . First, we identify the maximal linear modules and parallel modules, and merge the vertices in each module into a single super-vertex. We add an edge from super-vertex  $s_1$  to super-vertex  $s_2$  if and only if there exists  $u \in s_1$ , and  $v \in s_2$  such that  $(u, v)$  is an edge in  $G$ . In this way, we obtain the first level compressed graph  $G_1 = \text{Compress}(G)$ . Clearly,  $G_1$  is also a DAG without redundant edges. Then we apply the same compression process to  $G_1$  to obtain the next level compressed graph  $G_2 = \text{Compress}(G_1)$ , and this process is repeated until we obtain a graph  $G_c$  which can no longer be compressed, i.e.,  $G_c$  does not have singleton-set parallel or linear modules.

*Example 1.* Consider the DAG  $G$  of Fig. 3(a), which consists of ten vertices numbered 1 to 10. The graph is reduced to  $\tilde{G}$  in Fig. 3(b) after transitive reduction. We will apply our compression to graph  $\tilde{G}$ .

There are no parallel modules in  $\tilde{G}$ . However, vertices 2, 3 and 4 can form a maximal linear module. Another maximal linear module exists in  $\tilde{G}$  that consists of vertices 5, 6 and 7. So, vertices 2, 3, 4 and vertices 5, 6, 7 are compressed into two single nodes, and they are reduced into nodes LS1 and LS2 respectively in graph  $G_1$  shown in Fig. 4(a) after the first level compression. Then  $G_1$  is compressed again to obtain  $G_2$  as shown in Fig. 4(b), where the nodes LS1, LS2 and 8 in  $G_1$  are merged as they form an equivalent set in  $G_1$ . The third level compression creates graph  $G_3$  in Fig. 4(c) by merging nodes 1 and IS1 in  $G_2$  which form a linear module. The graph  $G_3$  does not contain any parallel or linear modules thus cannot be compressed further. So,  $G_3$  is the final compressed graph of data graph  $G$ .



**Fig. 4.** (a) Graph  $G_1$  after first level compression, (b) Graph  $G_2$  after second level compression and (c) Graph  $G_3$  after final compression. LS denotes linear module and IS denotes parallel module

We organize the modules in all levels of the compressed graphs into a tree structure, called the *modular decomposition tree*, or *decomposition tree* for brevity, as follows: The root of the tree is the final compressed graph  $G_c$ . Each module in the previous-level compressed graph  $G_{c-1}$  is a child node of the root; Each child node of the root that corresponds to a non-trivial module of  $G_{c-1}$ ,

in turn, has its own children, representing modules in the previous level graph  $G_{c-2}$ . This continues until we reach the nodes representing modules in the first-level compressed graph, where each non-trivial module points to their children, which are individual vertices in the original graph  $G$ . Note that the leaf nodes of the tree are individual vertices in the original graph  $G$ . Also, to help reachability detection, we keep a record of the vertex positions in the chain of each linear module in a compressed graph  $G_i$ , where if the starting vertex has position 1, the next vertex will have position 2, and so on. We will use  $pos(v, LS)$  to denote the position of node  $v$  in the chain of  $LS$ . Obviously, for  $u, v \in LS$ ,  $u \rightsquigarrow_{G_i} v$  if and only if  $pos(u, LS) < pos(v, LS)$ .

Figure 5 shows the modular decomposition tree,  $T$ , of graph  $\tilde{G}$  in Fig. 3(b). Let  $M$  be a non-leaf node in the decomposition tree of  $G$ . By definition,  $M$  is either a parallel or linear module in some compressed graph  $G_i$  ( $i < c$ ), or it is the final compressed graph  $G_c$ .  $M$  can be regarded as a set of the original vertices of  $G$  in the obvious way. Put in another way, we say vertex  $v \in G$  belongs to (or is in)  $M$  if  $v$  is a descendant of  $M$  in the decomposition tree. For example, the vertices 2, 3, 4, 5, 6, 7, 8 belong to the module IS1 in Fig. 5.

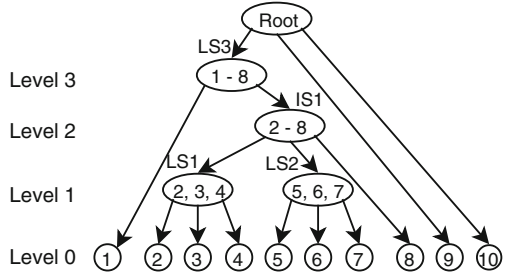


Fig. 5. The modular decomposition tree  $T$  of graph  $\tilde{G}$

We have the following observations about modules in the decomposition tree:

**Lemma 5.** *The vertices of  $G$  that belong to each parallel or linear module  $M$  in  $G_i$  ( $i < c$ ) form a module of  $G$ . In other words, all vertices in  $M$  have the same external ancestors, as well as the same external descendants in  $G$ .*

The above lemma can be easily proved by induction on the compression level  $i$ . Using Lemma 5, we can easily see:

**Lemma 6.** *Given two distinct nodes  $N_1$  and  $N_2$  in  $G_i$  ( $i \leq c$ ),  $N_1 \rightsquigarrow_{G_i} N_2$  iff  $u \rightsquigarrow_G v$  for every pair of vertices  $u \in N_1, v \in N_2$ .*

**Answering Reachability Queries Using Modular Decomposition Tree.**

Suppose we have the decomposition tree  $T$  of  $G$ . For ease of presentation, let us use  $G_0$  to denote the graph  $G$ . For any pair of vertices  $u, v$  in the original graph  $G$ , we use  $LCA(u, v)$  to denote the lowest common ancestor of  $u$  and  $v$  in  $T$ . Note that  $LCA(u, v)$  corresponds either to a module in some compressed graph  $G_i$  ( $i \in [1, c - 1]$ ), or to the final compressed graph  $G_c$  (i.e., the root of  $T$ ). We have the following result.

**Theorem 1.** *Given two vertices  $u, v \in V_G$ , if  $LCA(u, v)$  corresponds to a parallel module of some graph  $G_i$  ( $i \in [1, c - 1]$ ), then  $u$  cannot reach  $v$  in  $G$ .*



*Proof.* If  $LCA(u, v)$  corresponds to a parallel module  $M$  of  $G_i$ , then suppose  $N_1$  and  $N_2$  are the two vertices of  $G_i$  that contain  $u$  and  $v$  respectively. By Lemma 4, we know  $N_1$  cannot reach  $N_2$  in  $G_i$ . Then by Lemma 6, we know  $u$  cannot reach  $v$  in  $G$ .

With the above discussion, we are ready to present the method for answering reachability queries using the decomposition tree and the final compressed graph  $G_c$ . Given two vertices  $u, v \in V_G$ , to find whether  $u \rightsquigarrow_G v$ , we can find the lowest common ancestor  $LCA(u, v)$  of  $u$  and  $v$ , and check the following:

1. If  $LCA(u, v)$  is a parallel module, then  $u$  cannot reach  $v$  in  $G$ , by Theorem 1.
2. If  $LCA(u, v)$  is a linear module, say  $M$ , then we check the positions of  $N_1$  and  $N_2$  in the corresponding chain of vertices in  $M$ , where  $N_1$  is the child of  $LCA(u, v)$  in the decomposition tree that contains  $u$ , and  $N_2$  is the child of  $LCA(u, v)$  that contains  $v$ , and  $u$  can reach  $v$  in  $G$  if and only if  $pos(N_1, M) < pos(N_2, M)$ .
3. If  $LCA(u, v)$  is the root of  $T$ , namely  $G_c$ , then suppose  $N_1, N_2$  are the children of  $LCA(u, v)$  that contain  $u$  and  $v$  respectively. Then  $u \rightsquigarrow_G v$  if and only if  $N_1 \rightsquigarrow_{G_c} N_2$ . Thus we only need to check whether  $N_1 \rightsquigarrow_{G_c} N_2$ . We can do it using any existing reachability algorithms. Since  $G_c$  is usually much smaller than  $G$ , checking  $N_1 \rightsquigarrow_{G_c} N_2$  in  $G_c$  is likely to be faster than checking  $u \rightsquigarrow_G v$  in  $G$ .

*Example 2.* Consider the decomposition tree  $T$  shown in Fig. 5.

- (1) For the query  $2 \rightsquigarrow_G 6$ , we find lowest common ancestor of vertices 2 and 6 is a parallel module, therefore, we know vertex 2 cannot reach vertex 6.
- (2) For the query  $2 \rightsquigarrow_G 4$ , we find  $LCA(2, 4)$  is a linear module, and the position of vertex 2 is before that of vertex 4. Therefore we conclude that  $2 \rightsquigarrow_G 4$ .
- (3) For the query  $2 \rightsquigarrow_G 9$ , Since 2 and 9 are in different children of the root, i.e.,  $LS3$  and 9 respectively, we only need to check whether  $LS3 \rightsquigarrow_{G_3} 9$ .

## 5 Algorithms

The previous section provides the main ideas of our approach. This section presents the detailed algorithms.

### 5.1 Building Modular Decomposition Tree

Algorithm 1 shows the process of creating the modular decomposition tree along with the final compressed graph. The algorithm takes a DAG that has no redundant edges  $G$  as input and returns the modular decomposition tree and the final compressed graph. The algorithm first creates a tree with a root node  $r$ . Starting with a random vertex  $v$ , the algorithm first tries to find all other vertices that can form a linear module with  $v$  (Line 7). If no such module is found then it will

**Algorithm 1.** BuildMDT( $G$ )

---

```

Input: DAG  $G$  with no redundant edges
Output: Modular Decomposition Tree  $T$  and  $G_c$ 
1 if  $T$  does not exist then
2    $\lfloor$  Create Tree  $T$  with root node  $r$ ;  $i \leftarrow 0$ ;  $G_i \leftarrow G$ 
3    $S \leftarrow \emptyset$ 
4   for each  $v \in V_{G_i}$  do
5     if  $v.isVisited$  is false then
6        $v.isVisited \leftarrow true$ 
7        $M \leftarrow FindLinearModule(v, G_i)$ 
8       if  $M \neq null$  then
9          $S \leftarrow S \cup M$ 
10        Add  $M$  as child of  $r$ 
11        for each vertex  $u$  in  $M$  do
12           $\lfloor$   $u.isVisited \leftarrow true$ ; Add  $u$  as a child of  $M$ 
13      else
14         $M \leftarrow FindParallelModule(v, G_i)$ 
15        if  $M \neq null$  then
16           $S \leftarrow S \cup M$ 
17          Add  $M$  as child of  $r$ 
18          for each vertex  $u$  in  $M$  do
19             $\lfloor$   $u.isVisited \leftarrow true$ ; Add  $u$  as a child of  $M$ 
20      else
21         $\lfloor$  Add  $v$  as a child of  $r$ 
22 if  $S \neq \emptyset$  then
23    $i++$ 
24    $G_i \leftarrow Compress(G_{i-1}, S)$ 
25   BuildMDT( $G_i$ )
26 else
27    $r \leftarrow G_i$ 
28    $\lfloor$  return  $T, G_i$ 

```

---

search for a maximal parallel module for  $v$  (Line 14). If such a module cannot be found, then  $v$  will be added as a child of  $r$  (Line 21), otherwise the found module  $M$  will be added as child of  $r$ , and each vertex in the module will be added as a child of  $M$  (Lines 8–12, 16–19). We record all such modules in  $S$  (Lines 9,16), and use them to compress the graph into a new graph (Line 24). Then we recursively call the algorithm to compress the new graph (Line 27). If no non-single-vertex module is found in the current graph, the current tree  $T$  will be returned, and the current graph will be returned as  $G_c$ .

The functions `FindParallelModule()` and `FindLinearModule()` used in Algorithm 1 are shown in Algorithms 2 and 3 respectively. These algorithms try to find a relevant module based on Lemma 3.

Algorithm 2 takes a vertex  $v$  and DAG  $G$  as input, and it finds the set of vertices,  $M$ , that share the same parents and same children with  $v$ . To do that, it first finds the set of vertices,  $M_1$ , that share the same parents with  $v$ , and then finds the set of vertices,  $M_2$ , that share the same children with  $v$ . Then  $M$  is the intersection of  $M_1$  and  $M_2$ .

Algorithm 3 takes a vertex  $v$  and DAG  $G$  as input, and it first searches for a possible chain of ancestors of  $v$  (Lines 2–12), and then searches for a chain of descendants of  $v$  (Lines 13–26). Both parts are via an iterative process.

Specifically, Lines 2 and 4 check whether  $v$  has a sole parent  $v'$ , and  $v'$  has a sole child  $v$ , if so  $v'$  is the parent of  $v$  in a linear module. After that lines 7–12 try to find a parent of the first vertex in the chain, one by one. In the process, it also provides a position number for each vertex found (Line 10). Note that the position does not have to be a positive number, as long as it can provide an appropriate order of the vertices in the chain, it will be fine. Lines 13–26 work similarly.

**Complexity.** Algorithm 3 takes  $L$  steps to find the linear module that contains  $v$  (checking the  $|parent(v)| = 1$  is just checking the in-degree of  $v$ ), where  $L$  is the size of the linear module that contains  $v$ . Algorithm 2 takes  $\sum_{u \in parent(v)} |child(u)| + \sum_{u \in child(v)} |parent(u)|$  steps to find vertices that share the same parents and same children with  $v$ . If we use  $I_{max}$  and  $O_{max}$  to denote the maximum in-degree and maximum out-degree respectively, then Algorithm 2 takes  $O(I_{max} \times O_{max})$  time. In Algorithm 1, for the first level compression, we visit each vertex in  $V_G$  that has not been put in a module, and once the vertex is visited or put into a module, it will no longer be visited. In the worst case, no non-trivial module exists, so that every vertex will be visited. Therefore, the first level compression takes  $O(|V| \times I_{max} \times O_{max})$ . Each next level compression will take no more than that of the previous level. Therefore, Algorithm 1 takes  $O(|V| \times I_{max} \times O_{max} \times h)$ , where  $h$  is the height of the decomposition tree.

## 5.2 Finding Reachability Using the Decomposition Tree

As discussed in the previous subsection, to answer reachability query  $?u \rightsquigarrow_G v$  using the decomposition tree, we only need to find  $LCA(u, v)$  and then take appropriate actions depending on the type of  $LCA(u, v)$ . To save time for finding the LCA, we design a slightly modified algorithms as shown in Algorithm 4. Given vertices  $u$  and  $v$ , we first find the children of the root that  $u$  and  $v$  belong

---

### Algorithm 2. FindParallelModule

---

**Input:** DAG  $G$  with no redundant edges, vertex  $v$

**Output:** The maximal nontrivial parallel module that  $v$  is in, or null if such module does not exist

```

1 Create module  $M = \{v\}$ 
2  $M.type = trivial$ 
3 if  $|parent(v)| = 0$  then
4    $M_1 \leftarrow \{v' \mid |parent(v')| = 0\}$ 
5 else
6    $M_1 \leftarrow \{v' \mid v' \in \bigcap_{u \in parent(v)} child(u), v' \neq v\}$ 
7 if  $|child(v)| = 0$  then
8    $M_2 \leftarrow \{v' \mid |child(v')| = 0\}$ 
9 else
10   $M_2 \leftarrow \{v' \mid v' \in \bigcap_{u \in child(v)} parent(u), v' \neq v\}$ 
11 if  $M_1 \cap M_2 \neq \emptyset$  then
12    $M.type \leftarrow Parallel\ M \leftarrow (M_1 \cap M_2) \cup M$  Return  $M$ 
13 else
14   Return null
```

---

to respectively, let us suppose  $u \in N_1$ ,  $v \in N_2$ , if they are different (this is equivalent to say  $LCA(u, v)$  is the root), we will use some existing algorithm to check  $?N_1 \rightsquigarrow_{G_c} N_2$ . Otherwise we will find the lowest *linear* LCA of  $u, v$  (note that to do so we only need to record the linear module ancestors of the vertices), if no such LCA exists, then  $u$  cannot reach  $v$ . Otherwise, suppose the linear LCA is  $M$ , we will check the relative positions of  $u$  and  $v$  in  $M$  to determine whether  $u$  can reach  $v$ . Here the position of  $u$  is defined to be same as the position of the child of  $M$  that contains  $u$ . If  $LCA(u, v)$  is a parallel module then that module will be a child of  $M$ . So,  $u$  and  $v$  will have the same position in  $M$  thus  $u$  cannot reach  $v$ . It is easy to see that the algorithm is equivalent to the process described in Sect. 4, hence its correctness is guaranteed.

**Size of the Decomposition Tree.** To answer reachability queries in the original graph  $G$ , we only need to store the final compressed graph  $G_c$  and the decomposition tree  $T$ . The total number of nodes in the tree is  $|V| + m + 1$ , where  $m$  denotes the number of non-trivial modules. The number of edges in  $T$  is  $|V| + m$ .

## 6 Experiments

**Setup:** We obtained the source code of DAG reduction, IP<sup>+</sup> and Feline from the authors which are written in C++, and compiled them using G++ 7.3.0. We implemented our multilevel compression and reachability query processing algorithms in C# using Visual Studio 2017. We created the index of IP<sup>+</sup> and Feline for each graph using the original code from the authors. Then we used that index to process the reachability queries. The experiments were run on a PC with Intel Core i7-7700 with 3.60 GHz CPU, 32 GB memory and Windows 10 operating system. We tested our approach with 10 real data sets. First we applied the transitive reduction of [21] to find  $\tilde{G}$ , which is a DAG without redundant edges. Then we applied our multilevel compression algorithm to get  $G_c$ . We used two state-of-the-art reachability algorithms IP<sup>+</sup> [18] and Feline [17] to process reachability queries over  $G_c$  and over the graph  $G^c$  which is obtained by DAG reduction. We compared our method with DAG reduction [21] which is the most recent graph compression method for reachability queries. We randomly generated 100,000 reachability queries for each graph.

**Datasets:** We used 10 real datasets Kegg<sup>1</sup>, XMark (see footnote 1), Patent (see footnote 1), Citeseerx (see footnote 1), soc-Epinions<sup>2</sup>, Web (see footnote 2), LJ (see footnote 2), 05Patent<sup>3</sup>, 05Citeseerx (see footnote 3) and DBpedia<sup>4</sup>.

<sup>1</sup> <https://code.google.com/archive/p/grail/downloads>.

<sup>2</sup> <http://snap.stanford.edu/data/index.html>.

<sup>3</sup> <http://pan.baidu.com/s/1bpHkFJx>.

<sup>4</sup> <http://pan.baidu.com/s/1c00Jq5E>.

**Algorithm 3.** FindLinearModule**Input:** DAG  $G$  with no redundant edges, vertex  $v \in V_G$ **Output:** The maximal nontrivial linear module that  $v$  is in, or null if such module does not exist

```

1 Create module  $M = \{ \}$ ;  $M.type = trivial$ 
2 if  $|parent(v)| = 1$  then
3    $v' \leftarrow$  unique parent of  $v$ 
4   if  $|child(v')| = 1$  /*  $v'$  and  $v$  are in the same linear module
5   then
6     add  $v, v'$  to  $M$ ;  $M.type \leftarrow Linear$ ;  $pos(v, M) \leftarrow 1$ ;  $pos(v', M) \leftarrow pos(v, M) - 1$ 
7     while  $|parent(v')| = 1$  do
8        $u \leftarrow$  unique parent of  $v'$ 
9       if  $|child(u)| = 1$  then
10        | add  $u$  to  $M$ ;  $pos(u, M) \leftarrow pos(v', M) - 1$ ;  $v' \leftarrow u$ 
11        | else
12        | | break
13 if  $|child(v)| = 1$  then
14    $v' \leftarrow$  unique child of  $v$ 
15   if  $|parent(v')| = 1$  then
16     if  $M.type = trivial$  then
17       | add  $v, v'$  to  $M$ ;  $M.type \leftarrow Linear$ 
18       |  $pos(v, M) \leftarrow 1$ ;  $pos(v', M) \leftarrow pos(v, M) + 1$ 
19     else
20       | add  $v'$  to  $M$ ;  $pos(v', M) \leftarrow pos(v, M) + 1$ 
21     while  $|child(v')| = 1$  do
22        $u \leftarrow$  unique child of  $v'$ 
23       if  $|child(u)| = 1$  then
24        | add  $u$  to  $M$ ;  $pos(u, M) \leftarrow pos(v', M) + 1$ ;  $v' \leftarrow u$ 
25        | else
26        | | break
27 if  $M.type = trivial$  then
28   | Return null
29 else
30   | Return  $M$ 

```

**Algorithm 4.** Find reachability from vertex  $u$  to vertex  $v$ **Input:** Modular decomposition tree  $T$ , Compressed Graph  $G_c$ , vertex  $u$ , vertex  $v$ **Output:** true if  $u$  is reachable to  $v$ , false otherwise

```

1  $N_1 \leftarrow$  Corresponding node of  $u$  in  $G_c$ 
2  $N_2 \leftarrow$  Corresponding node of  $v$  in  $G_c$ 
3 if  $N_1 = N_2$  then
4    $M \leftarrow FindLinearLCA(u, v)$ 
5   if  $M$  exists then
6     | if  $pos(u, M) < pos(v, M)$  then
7     | | return true
8   | return false
9 else
10  | return AlgoReachability( $G_c, N_1, N_2$ )

```

**Table 1.** Datasets and their compression ratio after ER reduction and multilevel compression.  $r_n(r_e)$  is the ratio of the number of vertices (edges) in  $\tilde{G}$ ,  $G^e$  and  $G_c$ 

Dataset	$G$		$\tilde{G}$	$G^e$		$G_c$	
	$ V $	$ E $	$r_e\%$	$r_n\%$	$r_e\%$	$r_n\%$	$r_e\%$
Kegg	3617	3908	93.8	37.6	35.7	9.7	9.3
XMark	6080	7025	99	55.8	57	25.7	31
soc-Epinions	42176	43797	96.6	19.9	19.3	13	12.7
Web	371764	517805	79.8	30.5	24.9	16.6	14.6
LJ	971232	1024140	95.1	11.1	10.8	7.9	7.6
Patent	3774768	16518947	71.6	91.2	68.9	90.5	68.7
05Patent	1671488	3303789	90.1	80.3	78.9	78.8	78.2
05Citeseerx	1457057	3002252	81	37.9	50	37.4	49.7
Citeseerx	6540401	15011260	74.4	39.7	46.4	38.9	46.1
DBpedia	3365623	7989191	59.2	50.5	31.7	43.9	28.9

**Table 2.** Graph size before and after compression

Dataset	$G$	$G^e$		$G_c$	
	$ V  +  E $	$ V_{G^e}  +  E_{G^e}  +  IS $	$r_{G^e}\%$	$ V_{G_c}  +  E_{G_c}  +  m $	$r_{G_c}\%$
Kegg	7,825	2,816	36	1,340	17.1
XMark	13,105	8,312	63.4	6,282	47.9
soc-Epinions	85,973	17,602	20.5	13,433	15.6
Web	889,569	265,341	29.8	193,252	21.7
LJ	1,995,372	226,830	11.4	180,308	9
Patent	20,293,715	15,011,351	74	14,986,127	73.8
05Patent	4,975,277	4,111,385	82.6	4,086,800	82.1
05Citeseerx	4,459,309	2,180,701	48.9	2,173,504	48.7
Citeseerx	21,551,661	10,110,673	46.9	10,058,232	46.7
DBpedia	11,354,814	4,517,284	39.8	4,226,247	32.2

Among the datasets Kegg and XMark are very small graphs. Datasets soc-Epinions, Web and LJ are comparatively larger whereas other 5 graphs can be considered as large graphs. Here Kegg is a metabolic network and XMark is an XML document. Datasets soc-Epinions and LJ are the online social networks. Web is the web graph from Google. Patent, 05Patent, 05Citeseerx and Citeseerx are the citation networks and DBpedia is a knowledge graph. The statistics of these datasets are shown in the first two columns of Table 1.

## 6.1 Comparison of Compression Ratio

The compression ratios of transitive reduction, DAG reduction (i.e., transitive reduction and equivalence reduction), and our modular decomposition are shown in Table 1. From the table we can see that our approach has more compression for every graph than DAG Reduction. The dataset XMark has the best result with 30.1% more compression of vertex and 26% more compression of edges than DAG Reduction. For larger graphs, DBpedia shows best compression with 6.6% more compression of nodes and 2.8% more compression of edges. On the other hand, our compression scheme does not show much better compression ratio than DAG reduction over the Citeseerx and the Patent data sets. This could be because these data sets do not contain many linear modules. Generally, the reduction ratio depends on the structure of the graph.

Note however, a small percentage of compression for a large graph can also have great impact on query processing since even a small percentage of compression means reduction of lots of vertices and edges in large graphs (see the Patent dataset in Table 6 for example).

Table 2 shows the size of  $G$ ,  $G^\epsilon$  and  $G_c$ . Here, we calculated the size of the graph as the sum of the number of vertices and the number of edges. For  $G^\epsilon$  we have also added the number of equivalent classes as we need them for reachability detection. For the same reason, we have added the number of modules to the size of graph  $G_c$ . Table 3 shows the time required for building the decomposition tree using our algorithms which are implemented in C#, where the dataset Dbpedia

**Table 3.** Compression time (sec.)

Dataset	Time (sec.)
Kegg	0.057
XMark	0.16
soc-Epinions	4.49
Web	286.67
LJ	3073
Patent	389.23
05Patent	71.65
05Citeseerx	175.69
Citeseerx	8772.81
Dbpedia	89764.32

**Table 4.** Index construction time (ms)

Dataset	IP <sup>+</sup>		Feline	
	$G^\epsilon$	$G_c$	$G^\epsilon$	$G_c$
Kegg	0	0	0.60	<b>0.49</b>
XMark	0.02	<b>0</b>	0.79	<b>0.56</b>
soc-Epinions	0.04	<b>0.03</b>	2.18	<b>1.77</b>
Web	0.04	<b>0.03</b>	47.21	<b>29.53</b>
LJ	0.05	<b>0.03</b>	35.98	<b>29.83</b>
Patent	5.7	<b>5.64</b>	3959.15	<b>3732.31</b>
05Patent	1.42	<b>1.31</b>	<b>1051.72</b>	1056.2
05Citeseerx	0.53	<b>0.51</b>	<b>336.61</b>	341.9
Citeseerx	3.16	<b>2.78</b>	1836.02	<b>1834.36</b>
DBpedia	1.24	<b>1.14</b>	874.81	<b>845.69</b>

**Table 5.** Index size (MB)

Dataset	IP <sup>+</sup>		Feline	
	$G^\epsilon$	$G_c$	$G^\epsilon$	$G_c$
Kegg	0.043	<b>0.008</b>	0.031	<b>0.008</b>
XMark	0.12	<b>0.05</b>	0.08	<b>0.03</b>
soc-Epinions	0.21	<b>0.12</b>	0.19	<b>0.12</b>
Web	3.25	<b>1.62</b>	2.59	<b>1.41</b>
LJ	2.71	<b>1.72</b>	2.47	<b>1.75</b>
Patent	67.91	<b>66.84</b>	78.76	<b>78.22</b>
05Patent	18.35	<b>17.79</b>	30.71	<b>30.16</b>
05Citeseerx	12.64	<b>12.47</b>	12.63	<b>12.47</b>
Citeseerx	67.91	<b>66.84</b>	59.46	<b>58.27</b>
DBpedia	45.35	<b>38.54</b>	38.87	<b>33.84</b>

has taken the most time and is comparable that of dataset Citeseerx. As the indexing is done offline, we consider these timings as viable in practice.

## 6.2 Performance on Reachability Query Processing

Table 4 shows the comparison of index construction time for  $IP^+$  and Feline algorithms over  $G^\epsilon$  and  $G_c$ . The better results are highlighted in bold font in the table. Here, multilevel compression requires less index construction time for every graph for creating index for  $IP^+$ . For Feline, we also have better result for each graph except 05Patent and 05Citeseerx. The index size of  $IP^+$  and Feline, for  $G^\epsilon$  and  $G_c$ , are shown in Table 5. From the table we can see that the index sizes of  $G_c$  are smaller for every graph than  $G^\epsilon$  for both  $IP^+$  and Feline,

although for the Citeseerx and Patent datasets the difference is very small. This is not surprising because the sizes of  $G_c$  and  $G^\epsilon$  are very close for these data sets.

Table 6 shows the comparison of the query time for both  $IP^+$  and Feline. We run each query 10 times and the time shown is the average of the 10 runs. We can see that our compression outperforms DAG reduction in query processing for almost every graph. Surprisingly, the best improvement is over the Patent dataset, where our approach is much faster than DAG reduction in both  $IP^+$  and Feline. It is also surprising that  $IP^+$  is lower using our approach than using DAG reduction in Kegg dataset.

## 7 Conclusion

We presented a method to compress a DAG that has no redundant edges, using two types of modules, to obtain a decomposition tree. We showed how to use the decomposition tree to answer reachability queries over the original graph. Experiments show that for many real-world graphs, our method can compress the graph to much smaller graphs than DAG reduction, and reachability queries can be answered faster, and the index size can be smaller as well.

**Acknowledgement.** This work is supported by Australian Research Council discovery grant DP130103051.

**Table 6.** Query time (ms)

Dataset	$IP^+$		Feline	
	$G^\epsilon$	$G_c$	$G^\epsilon$	$G_c$
Kegg	<b>100</b>	116	26	<b>23</b>
XMark	165	<b>121</b>	45	<b>31</b>
soc-Epinions	108	<b>83</b>	30	<b>28</b>
Web	198	<b>186</b>	69	<b>65</b>
LJ	160	<b>151</b>	76	<b>72</b>
Patent	6866	<b>5414</b>	16459	<b>14186</b>
05Patent	189	189	134	<b>113</b>
05Citeseerx	315	<b>284</b>	169	<b>163</b>
Citeseerx	252	<b>246</b>	<b>121</b>	122
DBpedia	172	<b>170</b>	83	<b>73</b>



## References

1. Agrawal, R., Borgida, A., Jagadish, H.V.: Efficient management of transitive relationships in large data and knowledge bases. In: SIGMOD, pp. 253–262 (1989)
2. Cai, J., Poon, C.K.: Path-hop: efficiently indexing large graphs for reachability queries. In: CIKM, pp. 119–128 (2010)
3. Chen, Y., Chen, Y.: An efficient algorithm for answering graph reachability queries. In: ICDE, pp. 893–902 (2008)
4. Cheng, J., Huang, S., Wu, H., Chen, Z., Fu, A.W.: TF-label: a topological-folding labeling scheme for reachability querying in a large graph. In: SIGMOD (2013)
5. Cohen, E., Halperin, E., Kaplan, H., Zwick, U.: Reachability and distance queries via 2-hop labels. In: SIAM, pp. 937–946 (2002)
6. Fan, W., Li, J., Wang, X., Wu, Y.: Query preserving graph compression. In: SIGMOD, pp. 157–168 (2012)
7. Jagadish, H.V.: A compression technique to materialize transitive closure. *TODS* **15**, 558–598 (1990)
8. Jin, R., Ruan, N., Dey, S., Yu, J.X.: SCARAB: scaling reachability computation on large graphs. In: SIGMOD, pp. 169–180 (2012)
9. Jin, R., Ruan, N., Xiang, Y., Wang, H.: Path-tree: an efficient reachability indexing scheme for large directed graphs. *TODS* **36**, 7 (2011)
10. Jin, R., Wang, G.: Simple, fast, and scalable reachability oracle. *PVLDB* **6**, 1978–1989 (2013)
11. Jin, R., Xiang, Y., Ruan, N., Fuhry, D.: 3-HOP: a high-compression indexing scheme for reachability query. In: SIGMOD, pp. 813–826 (2009)
12. Jin, R., Xiang, Y., Ruan, N., Wang, H.: Efficiently answering reachability queries on very large directed graphs. In: SIGMOD, pp. 595–608 (2008)
13. McConnell, R.M., de Montgolfier, F.: Linear-time modular decomposition of directed graphs. *Discret. Appl. Math.* **145**(2), 198–209 (2005)
14. Seufert, S., Anand, A., Bedathur, S.J., Weikum, G.: Ferrari: Flexible and efficient reachability range assignment for graph indexing. In: ICDE, pp. 1009–1020 (2013)
15. Su, J., Zhu, Q., Wei, H., Yu, J.X.: Reachability querying: can it be even faster? *TKDE* **29**, 683–697 (2017)
16. Trißl, S., Leser, U.: Fast and practical indexing and querying of very large graphs. In: SIGMOD, pp. 845–856 (2007)
17. Veloso, R.R., Cerf, L., Junior, W.M., Zaki, M.J.: Reachability queries in very large graphs: a fast refined online search approach. In: EDBT, pp. 511–522 (2014)
18. Wei, H., Yu, J.X., Lu, C., Jin, R.: Reachability querying: an independent permutation labeling approach. *PVLDB* **7**, 1191–1202 (2014)
19. Yano, Y., Akiba, T., Iwata, Y., Yoshida, Y.: Fast and scalable reachability queries on graphs by pruned labeling with landmarks and paths. In: CIKM (2013)
20. Yildirim, H., Chaoji, V., Zaki, M.J.: GRAIL: a scalable index for reachability queries in very large graphs. *VLDBJ* **21**, 509–534 (2012)
21. Zhou, J., Zhou, S., Yu, J.X., Wei, H., Chen, Z., Tang, X.: DAG reduction: fast answering reachability queries. In: SIGMOD, pp. 375–390 (2017)



# Multiple Privacy Regimes Mechanism for Local Differential Privacy

Yutong Ye<sup>1,3</sup>, Min Zhang<sup>1,2</sup>, Dengguo Feng<sup>2(✉)</sup>, Hao Li<sup>1</sup>, and Jialin Chi<sup>1</sup>

<sup>1</sup> Trusted Computing and Information Assurance Laboratory, Institute of Software,  
Chinese Academy of Sciences, Beijing, China

{yeyutong,mzhang,lihao,chijialin}@is.iscas.ac.cn

<sup>2</sup> State Key Laboratory of Computer Science, Institute of Software,  
Chinese Academy of Sciences, Beijing, China

feng@is.iscas.ac.cn

<sup>3</sup> University of Chinese Academy of Sciences, Beijing, China

**Abstract.** Local differential privacy (LDP), as a state-of-the-art privacy notion, enables users to share protected data safely while the private real data never leaves user's device. The privacy regime is one of the critical parameters balancing between the correctness of the statistical result and the level of user's privacy. In the majority of current work, authors assume that the privacy regime is totally determined by the service provider and dispatched to all users. However, it is inelegant and unpromising for all users to accept the same privacy level in real world. In this paper, we propose a new LDP estimation method MLE which is applicable for the scenario of multiple privacy regimes. MLE uses the idea of parameter estimation to merge the results generated by users of different privacy levels. We also propose an extension of MLE to handle the situation when all users' regimes are in a continuous distribution. We also provide an Adapt estimator which assigns users to use different LDP schemes based on their regimes, and it performs better than the estimator with only one fixed LDP scheme. Experiments show that our methods provide a higher level of accuracy than previous proposals in this multiple regimes scenario.

**Keywords:** Local differential privacy · Multiple privacy regimes · Frequency estimation

## 1 Introduction

With the rapid penetration of Internet and Smartphone through the crowded, large-scale collection of user data is already a necessary daily activity for companies. User data has become one of the most important asset, which can give support to data scientists to discover new patterns and provide training examples for machine learning models. However, this comes with huge risks—can these companies protect users' sensitive data from malicious access? Disclosure may violate the users' privacy and lead to scandal.

Anonymization techniques are one common method to protect user privacy by blurring the personalized or identifiable information, but it's vulnerable to the de-anonymization attack as shown in the case of Netflix Price [15]. For the above scenario, Differential Privacy [6, 7] successfully achieved releasing sanitized datasets, but not at the client level.

Local Differential Privacy (LDP) [13] is a branch of DP, which gives the benefits of population-level statistics without the collection of raw private data. With LDP, service provider can get statistical information on all users by just aggregating users' noised report. This feature makes LDP widely used in real-world scenarios. For example, Google's use LDP scheme RAPPOR to constantly collect the home-page that all users like to set up; Apple announced its implementation of LDP in iOS 10 and MacOS in WWDC 2016; Microsoft also deploys a LDP-enabled data collection mechanism in Windows Insiders program to collect application usage statistics.

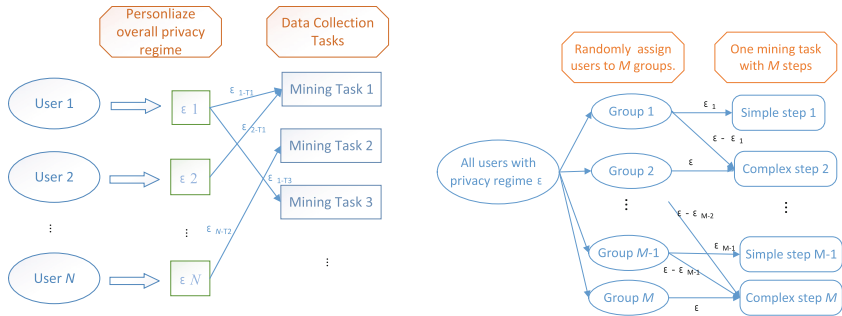
LDP's security parameter (privacy regime  $\epsilon$ ) represents the security level of its randomization process. The bigger (or smaller) the security parameter, the more (or less) availability of the noisy report that the user shares. However, most LDP schemes assume that each user has the same  $\epsilon$ , hence each user uses the exact same randomized procedure to generate a noisy report from their own data. There have been complaints that deployed LDP schemes use higher values of  $\epsilon$  while users are not given any choice. So it is questionable that  $\epsilon$  is entirely determined by the service provider who wants more availability of the data.

**Multiple and Personalized Privacy Regimes.** In order to meet the privacy demands of different people, we argue that users should be allowed to set their overall privacy levels (e.g., low/moderate/high) independently. Here, we assume that this personalization of privacy regimes does not mean the user should set a new  $\epsilon$  every time he shares data. Instead, users will set an infrequently changing  $\epsilon$ , which will be consumed a fixed percentage every time users share data with LDP. This assumption is derived from a study [17], which suggests that Apple's deployment [2] for LDP has an overall privacy regime as high as 16 everyday and sets privacy consumption to 1 or 2 each time shares data while there is no transparency.

In this paper, we consider that simple LDP mining task only contains one data collection activity, and the common simple mining task includes frequency estimation, mean value estimation, heavy-hitters identification and so on. So even if users have the same overall privacy regime  $\epsilon$ , multiple  $\epsilon$  may still appear in one mining task because the number of times that data is shared is different. For example, it could involve the real-time sharing of trajectory data and is not the focus of our analysis. We present the above personalization process in Fig. 1(a).

In addition, when LDP handles complex mining task like "Frequent Itemset Mining" [20] which contains several rounds of data collection activities, it is common to randomly assign all users to several groups, and then different groups finish each step of the mining task respectively with the same  $\epsilon$ . However, some

groups of users only need to pay a small amount of privacy to complete easy step (e.g., frequency estimation in small candidate space), the remaining can be assigned to challenging steps by segmentation as shown in Fig. 1(b). Combining the above two points, it's urgent to deal with multiple privacy regimes under LDP.



(a) Users set privacy regimes independently and the collection activity that each user may participate in is different. (b) Multiple regimes in complex mining tasks, users participate in different steps with different regimes

**Fig. 1.** Application scenarios for multiple privacy regimes.

In this paper, we assume that users may have different privacy acceptances for personalization: Once a user sets his overall privacy regime  $\epsilon$ , he does not change this value frequently and his participation in any tasks will automatically consume a certain proportion of this  $\epsilon$  until it's used up; Since the collection behavior is usually long-term and the user's privacy data may change (e.g., web pages visited), users' specific choices of privacy regimes are not related to the value of the private data.

As far as we know, frequency estimation is the most basic LDP mining task, so it is meaningful to apply it under the mechanism of multiple privacy regimes. An obvious method of frequency estimation in this scenario, is to divide users with the same  $\epsilon$  into the same group, and estimate frequency for different groups separately. In the end, service provider aggregates each group's estimated value by weighting. This is discussed in Sect. 3.

**Contributions**

- We propose MLE method which applies the idea of parameter estimation to obtain an optimal estimate from user groups with different privacy regimes. Our theoretical analysis shows that the accuracy of MLE can be equivalent to tradition method which forces all users to choose one specific privacy regime, and this equivalence shows that MLE is practical.

- We propose S-MLE method to handle the situation when all users' privacy regimes are subject to continuous distribution in one mining task. It uses a predefined  $\beta$  parameter to segment the contiguous  $\epsilon$  on the basis of MLE. The experiments show that the  $\beta$  parameter of S-MLE may greatly influences the accuracy under certain conditions.
- We propose Adapt-MLE method which encompasses different LDP schemes for multiple privacy regimes. And the performance of Adapt-MLE is better than that of MLE, especially when the discrete values of  $\epsilon$  of different users span a wide range.

## 2 Preliminaries and Notations

We assume that all users are willing to share their information to help service provider update its statistical information. For the sake of privacy, each user perturbs his own data by advanced technique (via LDP) with different demands for security. The service provider aims to find out the frequencies of values among the population. Such a process involves the following preliminaries.

### 2.1 Local Differential Privacy

**Definition 1** (*Local Differential Privacy*). An algorithm  $A$  satisfies  $\epsilon$ -local differential privacy ( $\epsilon$ -LDP) where  $\epsilon > 0$ , if and only if for any input  $v_1$  and  $v_2$ , we have  $\forall y \in \text{Range}(A)$ ,

$$\frac{\Pr(A(v_1) \in y)}{\Pr(A(v_2) \in y)} \leq e^\epsilon,$$

where  $\text{Range}(A)$  denotes the set of all possible outputs of the algorithm  $A$ .

When privacy regime  $\epsilon$  is small, the adversary can't identify the true value from the noise version reliably. The basic core of algorithm  $A$  is Randomized response (RR) [21], which is a statistical technique used for collecting social embarrassing questions.

### 2.2 Frequency Estimation in LDP

Let  $\mathbf{f} = (f_1, \dots, f_k)$  be a probability distribution on a set containing  $k$  candidate values,  $f_1$  is the true frequency of  $v_1$  and the sum of  $\mathbf{f}$  is 1. We can consider that  $\mathbf{f}$  is the proportion of  $N$  users choosing different values. Using LDP allows the server to obtain an estimate  $\hat{\mathbf{f}} = (\hat{f}_1, \dots, \hat{f}_k)$  without obtaining the user's original data.

Each of  $N$  users holds a value  $v_j$  taken from the above  $k$  candidates and shares this  $v_j$  to the service provider in a LDP manner. In the beginning, user need to encode the  $v_j$  to a specific format,  $x_j = E(v_j)$ , then select a parameter  $\epsilon$  to obtain  $y_j$  by randomization. Finally, service provider aggregates  $N$  data records  $(y_1, \dots, y_N)$  and figures out  $\hat{\mathbf{f}}$ .

### 2.3 General LDP Schemes

Randomization mechanisms that satisfy LDP have been widely studied in recent years, our work involves the following very common schemes.

**kRR** (k-ary Randomized Response) [12] is a generalization of binary randomized response (RR) which performs well in low privacy level.

**Base RAPPOR** is simplest configuration of RAPPOR [9] which has been widely accepted. It has a output alphabet  $v = \{0, 1\}^k$  of size  $2^k$ . It first maps  $v_i (1 \leq i \leq k)$  onto  $e_i \in 0, 1^k$ , where  $x = e_i$  is the  $i$ -th standard basis vector. At last a length- $k$  binary vector  $y$  is generated from  $x$  by  $y = RR(x)$ ,  $RR$  here can be seen with a pair of alterable probability  $p$  and  $q$ .

$$Pr(y[i] = 1|x[i] = 1) = p; \quad Pr(y[i] = 1|x[i] = 0) = q \tag{1}$$

Base Rappor sets  $p$  to  $e^{\epsilon/2}/(e^{\epsilon/2} + 1)$  and  $q$  to  $1 - p$ . For the fact that Base RAPPOR is classical and is the basis of many other schemes, we mainly use it to analyze the scenario of multiple privacy regimes in Sects. 3 and 4.

**Optimal Scheme [22] and OLH [18]** are two similar schemes, and they are all obtained by optimizing the probability of  $RR$  in the Base RAPPOR (Change  $p, q$  in Eq. 1), the difference is that the latter is evolved from SH [3] and reduces communication cost by hash method.

At Last, frequency estimation formulas of these schemes are all

$$\hat{f}_i = \frac{C(i) - q * N}{(p - q) * N}$$

where  $C(i)$  is the count of reported vector which has the  $i$ 'th bit being 1. From OLH, we get the following properties.

**Lemma 1.** *For LDP scheme which uses RR with probability  $p$  and  $q$ , the frequency estimation  $\hat{f}_i$  is an unbiased estimate of  $f_i$ , and its variance is*

$$var(\hat{f}_i) = \frac{(f_i * p + (1 - f_i) * q) * (1 - f_i * p - (1 - f_i) * q)}{N(p - q)^2}$$

Employing Base rappor's settings for  $p$  and  $q$  and taking  $e^{\epsilon/2} > 1 \gg f_i$  into account, Base rappor's variance is as follows:

$$var(\hat{f}_i) = \frac{e^{\epsilon/2}}{N(e^{\epsilon/2} - 1)^2} \tag{2}$$

## 3 Problem Formulation

In this paper, we consider the specific LDP problem that users specific choices of privacy level will lead to multiple  $\epsilon$  in one mining task. In this section, we first

**Table 1.** Notations

Notations	Explanations	Notations	Explanations
$v$	Raw data in frequency estimation	$p, q$	Defined in equation (1)
$x$	Encoded data	$N$	Total number of users in the collection
$y$	Sanitized data	$n_m$	The size of the $m$ -th group
$\mathbf{f}$	Frequency distribution	$k$	The domain of value
$f_i$	The true frequency of $v_i$	$\hat{f}_i$	An estimate of $f_i$

introduce a primary method called Raw-PCE (Personalized Count Estimation) which can be considered a simplified version of PCE [5] proposed for the handling spatial data aggregation. Compared with the original PCE scheme, only the customization of privacy regime is preserved in Raw-PCE while the customization of other factors are omitted. Then we analyze the estimate of Raw-PCE and get a new probability model (Table 1).

### 3.1 A Multiple Privacy Regime Scheme: Raw-PCE

**Raw-PCE** is an intuitive and primitive method handling this scenario. Suppose there are totally  $M$  different privacy regimes namely  $\epsilon_1, \epsilon_2, \dots, \epsilon_M$ .

Firstly, the service provider groups the users according to their personalized privacy regimes which results in totally  $M$  groups.

Then, each group with the same privacy regime independently generate its frequency estimate vector, the estimate vector generated by group  $m$  is denoted as  $\mathbf{f}(\hat{m}) = (f(\hat{m})_1, f(\hat{m})_2, \dots, f(\hat{m})_k)$ . Totally  $M$  estimates are generated. Without loss of generality, here we consider only one candidate value  $v$  to simplify the problem. The  $M$  estimates for value  $v$  can be denoted as  $f_{(1)}, f_{(2)}, \dots, f_{(M)}$ .

Finally, if there is no other auxiliary information, the way in which the estimated value  $\hat{f}$  is calculated by Raw-PCE is as follows:

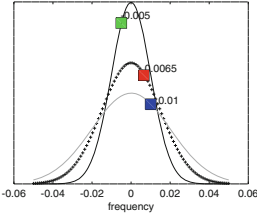
$$\hat{f} = \sum_{m=1}^M f_{(m)} \alpha_{(m)} \tag{3}$$

where  $\alpha(\cdot)$  represents the weights of each group’s estimation. Raw-PCE here ignores the fact that every estimate’s accuracy is different, and it just takes the size of each group as the weights where  $\alpha_m = n_m/N$ . Combining Eqs. 2 and 3, we have:

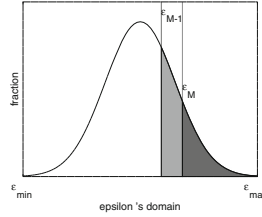
**Lemma 2.** *In Base RAPPOR scheme, estimation of Raw-PCE has the variance as follows,*

$$\text{var}(\hat{f}) = \sum_{m=1}^M \frac{n_m^2 e^{\epsilon_m/2}}{(e^{\epsilon_m/2} - 1)^2 * N^3}$$

The above formula shows that the error is cumulative, whenever there is an estimation with large error among  $M$  estimations, final errors are greatly increased which is rather unacceptable.



**Fig. 2.** The true frequency of  $v_i (i \in [k])$  is 0, and there are three estimations of  $f_i$  which are drawn from three normal distribution.



**Fig. 3.** S-MLE: dividing continuous  $\epsilon$  into discrete values and the size of each shadow area is  $\beta$ .

### 3.2 Probabilistic Model of Multiple Regimes Setting

Although users are divided into different groups with different  $\epsilon$ , their choices of privacy regimes can be considered as irrelevant to the choices of their favourite items, which means user’s possibility of choosing item  $v$  is the same in all  $M$  groups.

After the randomization process, the reported number  $C$  ( $C(i)$  is introduced in Sect. 2.3 and here we omit  $i$ ) is a random variable from a binomial distribution, namely  $C \sim \mathbf{B}(N, pf + q(1 - f))$ . We use  $p'$  to denote  $pf + q(1 - f)$ . Furthermore,  $N$  is usually big enough to ensure normal approximation and we have  $C \sim \mathbf{N}(Np', Np'(1 - p'))$ .  $f_{(m)}$  is a normalization of  $C_{(m)}$  and follows a Gaussian distribution.

Then all  $M$  groups follow  $M$  different normal distributions, which share the same expectations but have different variances due to the different values of  $\epsilon$ . Therefore, as Fig. 2 shows, the  $M$  estimates generated by  $M$  groups, respectively  $f_{(1)}, f_{(2)}, \dots, f_{(M)}$ , can be regarded as  $M$  random samplings of the actual user proportion  $f_v$ , each of which follows a unique normal distribution.

The problem of multiple privacy regimes can be regarded as equivalent to the problem of obtaining the best estimate of  $f_v$  from  $M$  group estimations  $f_{(1)}, f_{(2)}, \dots, f_{(M)}$ , which are random samples separately drawn from  $M$  normal distributions. Our target is simplified as to give the optimal estimate of the expectation with the help of parameter estimation methods.

## 4 Estimate Methods

In this section, we present two types of frequency estimate for multiple privacy regimes. In Sect. 4.1, we discuss a new MLE method and give theoretical proof.



Then we prove MLE’s accuracy is much better than the basic Raw-PCE (given in Sect. 3.1). In Sect. 4.2, we discuss the situation when there exists large number of groups after grouping operation and propose S-MLE method on the basis of MLE.

### 4.1 MLE

Maximum likelihood is an effective method in parameter estimation, which is adopted here to generate unbiased expectation  $f$ . Once the service provider gets  $M$  estimations  $(\hat{f}_{(1)}, \hat{f}_{(2)}, \dots, \hat{f}_{(M)})$  and their variances as well, the Maximum likelihood estimation (MLE)  $\hat{f}$  is defined as follows:

**Theorem 1.** *Given the  $M$  estimate  $(\hat{f}_{(1)}, \hat{f}_{(2)}, \dots, \hat{f}_{(M)})$ , the MLE for the multiple privacy regimes scenario is*

$$\hat{f} = \left( \sum_{m=1}^M \frac{\hat{f}_{(m)}}{\text{var}(\hat{f}_{(m)})} \right) / \left( \sum_{m=1}^M \frac{1}{\text{var}(\hat{f}_{(m)})} \right)$$

The proof is in Appendix A.

It’s interesting to observe that the expectation we derived for MLE is in a weighted-sum manner. The weights become the reciprocal of the variances. Bring it to Eq. 3 for demonstration,

$$\alpha_m = \frac{1/\text{var}(\hat{f}_{(m)})}{\sum_{m=1}^M 1/\text{var}(\hat{f}_{(m)})}$$

**Estimation Accuracy Analysis.** Due to MLE is unbiased and contains grouping operation, its accuracy can be somehow equivalent to a traditional LDP method with a special privacy regime  $\epsilon$ . Assuming there are two service providers doing the same collection on a population. One allows user to choose different  $\epsilon$  and groups them, finally there are  $M$  groups whose size and privacy regime are  $(n_1, \epsilon_1), (n_2, \epsilon_2), \dots, (n_M, \epsilon_M)$ ; The other makes all users in the same privacy regime. So in which condition they achieve the same level of accuracy or their estimations have the same variance. The latter obliges all users to have the same  $\epsilon$ , and here we call this reckless method traditional estimation (TE).

**Theorem 2.** *The variance of the estimates obtained by the MLE is  $\text{var}(\hat{f}) = 1/(\sum_{m=1}^M \frac{1}{\text{var}(\hat{f}_{(m)})})$ .*

The proof is in Appendix B. It can be inferred from the formula that if the data collector only uses the estimate with the least error, its effect is not as good as that of MLE which combines all the estimates.

When substituting the variance generated by Base RAPPOR into the Theorem 2, we obtain a new lemma.

**Lemma 3.** *In Base RAPPOR, if any  $e^{\epsilon_m/2} \gg 1$ , there exists a privacy regime  $\epsilon' \approx 2 * \ln \frac{\sum^{n_m * \exp(\epsilon_m)} n_m}{\sum^{n_m}}$  such that makes the accuracy of directly using TE method with  $\epsilon'$  equals to MLE with multiple  $\epsilon_m (m \in [M])$ .*

The proof is in Appendix C.

Comparing Theorem 2 with Lemma 2, we find that the overall variance of MLE is much lower than that of Raw-PCE method. The explanation can be that Theorem 2 implies the final estimate will be accurate as long as at least one of the estimates has low variance, while Lemma 2 has high variance if just one group has high variance. Our experiments also show MLE is much more accurate than Raw-PCE.

### 4.2 S-MLE

When the user are allowed to choose any value as their overall privacy regimes from a considerate large set, there will be too many groups and some groups inevitably contain too few users. In this scenario, the above MLE method may be inapplicable because large errors are introduced into  $\hat{f}$ . And one extreme situation is that  $\epsilon$  is continuous function over real number field as shown in Fig. 4. In this section, we start by analyzing why the minimum size of the group ( $\beta N$ ) should be set and explaining what factors will affect the value, then we provide a supplement method called S-MLE for this scenario.

**$\beta N$ : Minimum Size of the Group.** From the perspective of sampling theory, the essence of grouping process in MLE is that the users are randomly sampled into  $M$  groups, and the frequency estimation result of each group is equivalent to the result of sampling scheme without replacement. And what we'll find is that the sample size of this  $M$  group is different and there may exist invalid sample group because sample survey with low sample size introduces lots of sampling error and would not represent the whole. Specifically, small group's unbiased estimation  $f_{i(m)}$  on value  $v_i$  differs greatly from the actual results of the population. Namely,  $|E(f_{i(m)}) - f_i| < \sigma$  can't hold where  $\sigma$  is tolerable sampling error.

So it makes sense to determine the minimum of the sample size which is also a basis for dividing the group. According to the sampling theory, the sample size is usually determined by the variation degree of the research object, the total number of samples and the demand for accuracy. In our MLE, sample size can be mainly determined by the size of candidate set ( $k$ ), the total number of users ( $N$ ), and the complexity of candidate set's frequency (the distribution of  $\mathbf{f}$ ). So we get the proposition as follows:

**Proposition 1.** *In MLE, for any group whose size does not exceed  $\beta N$ , it's necessary to ignore its estimation or make users in this group join other higher privacy level group.*

Combining the Proposition 1 and MLE, we need to figure out an empirical value for  $\beta$  and segment privacy regimes and re-merge existing groups.

**S-MLE.** The S-MLE method can be further extended to the situation when the type of  $\epsilon$  value is unlimited (continuous distribution).

From Theorem 2 we can see that the entire accuracy is depend on all users' distribution of  $\epsilon$ . let users who have temp largest  $\epsilon$  value form a group (size =  $\beta * N$ ) recursively is an efficient segmenting way (segmenting in Fig. 3). Since it's difficult and complexity to figure out sample size  $\beta$ , we give some empirical values here. In the experiment when fixing  $N$  to 100000, we find  $\beta \in [0.05, 0.1]$  ( $[0.1, 0.15]$ ) is reasonable for the situation when  $\mathbf{f}$  is generated by zipf's distribution (uniform distribution) and  $k$  is ranging from 20 to 200,  $\beta$  should be bigger as  $k$  increases.

## 5 Adapt MLE and Universality of Mining Scenarios

MLE and S-MLE have been able to achieve relatively high accuracy in frequency estimation by Base RAPPOR. Furthermore, they are also applicable for other mining scenarios like heavy-hitter identification, and replace Base RAPPOR with other scheme for better accuracy. In this section, we propose the Adapt MLE method which adaptively selects the most suitable LDP scheme for each group of users to share data, then we briefly show how to apply MLE to other LDP mining tasks.

**Adapt MLE.** The process of selecting schemes just fits in with some work [11, 18, 22] on how to select LDP schemes based on privacy regime  $\epsilon$ , the size of  $k$  and communication cost.

Here we attach importance to accuracy and use variance as the evaluation criteria to select LDP scheme for frequency estimation. Because uniform distribution is the most difficult to estimate analyzed by minimax [22], we set each value in  $\mathbf{f}$  to be the same and easily calculate the variance of each scheme through Lemma 1. So by comparing their variances, we can get the following directly:

$$\text{Adapt MLE}(\text{Group } m) = \begin{cases} kRR & \text{if } k < 3e^{\epsilon_m} + 2 \\ \text{OptimalSchemes} & \text{otherwise} \end{cases}$$

In other scenarios like sampling step in frequent item mining [20], “ $3e^{\epsilon} + 2$ ” might change slightly. So the accuracy of Adapt MLE is higher than that of MLE when the discrete values of  $\epsilon$  of different users span a wide range. However, accuracy is not the only factor that matters, communication cost and computational complexity are also worth considering in real world.

As far as we know, our multiple privacy setting still can be used for heavy-hitters identification. SH [3] consists of two important steps. First step uses hash function to separate the values into a lot of channels, with high probability each channel has at most one frequent value, then identify whether there is a frequent item in each channel. Referred to Chen [5], multiple privacy setting is fully applicable to this step. The second step employs a frequency oracle to estimate the frequency of those frequent values obtained from first step. And this

is similar to the mining task of frequency estimation while obtaining variance from frequency oracle is in another form and the final estimates are slightly biased (still consistent with the goal of heavy-hitters).

## 6 Experiments

In this section, we evaluate and compare the performance of our proposed personalized approach through extensive experiments. Since there is no existing work for our settings, we mainly verify the correctness of our analysis.

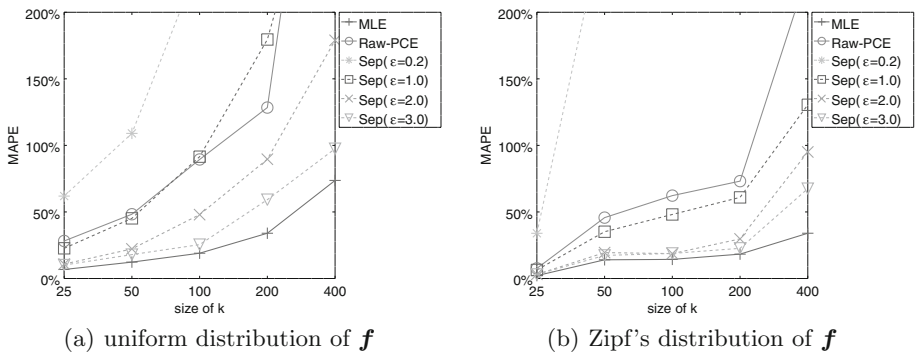
**Setup.** All experiments are performed 10 times and we plot the Mean Absolute Percentage Error (MAPE) of all frequency estimation. The MAPE is  $\frac{1}{k} \sum_i in[k] \frac{|f_i - \hat{f}_i|}{f_i + \sigma}$ , where  $f_i$  is the actual fraction of all users taking value  $i$  and  $\sigma$  is to prevent the denominator from 0.

In RAPPOR [9] with  $h = 1$ , their value for epsilon is actually set to  $2 \ln 3$  and the number of users is a million level. And Apple also set epsilon to 1 or 2. So we assume that 100 thousand users participates the collection, and set  $M$  options in most experiments to  $\epsilon = [0.2, 1.0, 2.0, 3.0]$  and the proportion of users in each group is  $G = [0.2, 0.3, 0.3, 0.2]$  where  $G$  denotes the corresponding proportion.

For better verification of correctness, we generate two synthetic data which are from Zipf’s distribution (parameter  $a = 2.5$ ) and uniform distribution. The schemes used in each experiment contain KRR, Base Rappor and Optimal Scheme. We change the distribution of  $\mathbf{f}$  by controlling the size of  $k$ .

### 6.1 Accuracy of MLE Method

Whatever the distribution of  $\mathbf{f}$  is in Fig. 4, MAPE curves generated by four groups shows that group with higher  $\epsilon$  generates better estimates; Raw-PCE’s accuracy has been greatly affected by the group with low variance, namely  $\epsilon =$



**Fig. 4.** Comparing MLE and Raw-PCE, varying  $k$ : estimates generated by each group are marked as Sep

0.2; From Theorem 2 and this figure, we know the variance of MLE is always smaller than the variance estimated by each group.

In uniform distribution (Fig. 4(a)), MAPE increases as  $k$  increases, roughly the same multiple because the denominator of MAPE’s calculation is actually  $1/k$ . In fact, the size of the variance is independent of  $k$ . So in zipf’s distribution (Fig. 4(a)), the change in MAPE will not be obvious when  $k$  does not exceed 200. This also the reason why uniform distribution is difficult to estimate.

### 6.2 S-MLE Method on Continuous $\epsilon$

When there are too many options of  $M$  or all users’  $\epsilon$  is continuously distributed, the value of  $\beta$  can help to divide all users into  $\lfloor 1/\beta \rfloor$  groups as described in Sect. 4.2. Small  $\beta$  will increase sampling error, but it can also reduce the overall variance from Theorem 2. A balance between overall variance and sampling error is reasonable.

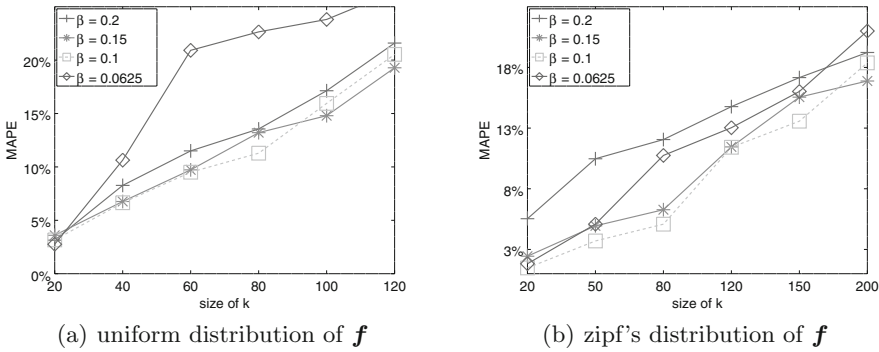


Fig. 5. the influence of  $\beta$  ’s value on MAPE, varying  $k$ .

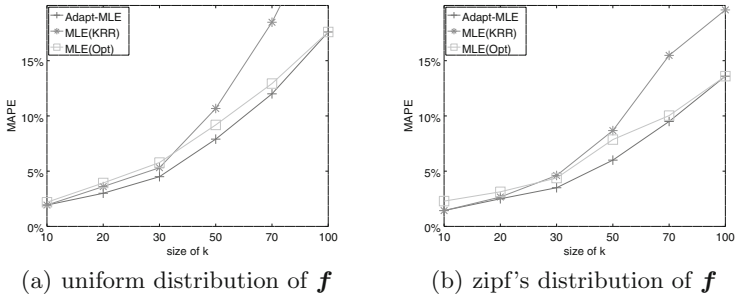
In this part, we make the proportion of users choosing different  $\epsilon$  obey  $N(2.5, 0.8)$  and  $\epsilon$ ’s contiguous interval be  $[1.0, 4.0]$ , namely  $G$  obeys  $N(2.5, 0.8)$  and  $\epsilon$  is continuous in  $[1.0, 4.0]$ . It can be observed from Fig. 5(a) that when  $k$  is small, the MAPE with small  $\beta$  is acceptable, namely, the sampling error has little effect. On the contrary, when  $k$  becomes larger than 60 for  $\beta = 0.0625$ , the sampling error even exceeds the error generated by estimator. But for Zipf’s distribution, the effect of  $k$  on sampling error is not obvious until  $k > 150$ . So for our settings above concerning the number of users and  $\epsilon$  ratio,  $\beta \in [0.05, 0.1]$  for Zipf’s distribution and  $\beta \in [0.1, 0.15]$  for uniform distribution are appropriate choices.

### 6.3 Multiple Schemes for Different Groups

In Sect. 5, we claim that users in different groups can use different LDP schemes to achieve better accuracy. From Sect. 5, kRR performs better when  $k < 3e^\epsilon + 2$ .

We divide 100 thousand users into 4 groups with  $\epsilon = [1.0, 2.0, 3.0, 3.2]$  and  $G = [0.3, 0.35, 0.3, 0.05]$ .

Learning from Fig. 6, “Adapt-MLE” performs better because users of these 4 groups use kRR if  $k$  are less than 10, 25 and 62 and 76 respectively and otherwise use Optimal Scheme.



**Fig. 6.** Adapt MLE with multiple schemes, “MLE(kRR)” represents only KRR and “MLE(Opt)” represents only Optimal scheme.

## 7 Related Work

The traditional differential privacy (DP) was developed for interactive query-response on a central database and provides theoretical privacy guarantee by mathematically randomizing the results of statistical queries. However, the major limitation of DP is that all users need to trust a central server. Despite attacks from aggregate queries, individual’s data may also suffer from privacy leakage before aggregation [8].

On the other hand, Local differential privacy (LDP) [13], a variant of DP, guarantees privacy of data without that server. Random Response (RR) [21], where the user responds either true or opposite answer depending on coin flipping, is the most basic technique in LDP schemes.

Widely accepted schemes for frequency estimation under LDP are Rappor by Erlingsson et al. [9] and succinct histogram (SH) by Bassily and Smith [3]. RAPPOR’s key idea is encoding values into Bloom filters and applying RR to each bit of Bloom filters. In order to conquer hash collision problems in Bloom filters, RAPPOR brings in cohorts. In this paper, we use RAPPOR’s no bloom filter version to analyze our multiple setting. SH’s has two important data structures—frequency oracle and succinct histogram, these two work together to estimate those values whose frequencies exceed  $\eta$ , so some applications do heavy hitters [5, 16, 19] identification referred to SH. Due to the high complexity of SH, Bassily et al. [4] recently developed an efficient way to query the frequency estimation based on SH mechanism.

In addition, discrete distribution estimation under LDP considers all values’ frequency accuracy. Kairouz et al. [11] analyzed several key factors (privacy

regimes, discrete distribution) which affect accuracy. Ye et al. [22] came up with Optimal Scheme for discrete distribution estimation. About [11, 22], their analysis tools all contained minimax with  $l_2$ -norm as loss function which is similar to variance. Wang et al. [18] introduced a framework that can generalize the most LDP schemes by recognizing RR's features. These work all have a prerequisite that the size of  $k$  is limited.

In personalization privacy fields, Jorgensen et al. [10] incorporated personalized settings for DP (PDP), and Li et al. [14] proposed a  $k$ -partition strategy to improve it. Then Chen et al. [5] first introduced the concept of personalized privacy in LDP (PLDP), it allows users to have two optional privacy regime  $\epsilon$  and  $\tau$ . The former has no change, while  $\tau$  represents a small piece of the candidates list. They assume some users set small size of candidates list and this part of users can greatly improve their performance on heavy-hitter mining task. Obviously,  $\tau$  makes this personalization process complex for users. Akter et al. [1] borrowed the definition of PLDP to estimate numeric data like average instead of heavy-hitters mining.

## 8 Conclusion

In this paper, we mainly study frequency estimation under Local Differential Privacy (LDP) in multiple regimes scenarios. We have formulated the problem of multiple privacy levels and proposed a MLE method to deal with this situation. Then, we propose S-MLE and Adapt-MLE to deal with the situation when users' privacy levels are in some special cases.

**Acknowledgments.** This work is supported by the National Natural Science Foundation of China (No. U1636216) and National Key R&D Program of China (No. 2016YFB0502302).

## 9 Appendix

### A. Proof of Theorem 1

*Proof.* ( $f_{(1)}^{\hat{}}$ ,  $f_{(2)}^{\hat{}}$ , ...,  $f_{(M)}^{\hat{}}$ ) are drawn from different normal distributions, normal distribution has probability density function as follows:

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma^2} \exp\left(-\frac{(x-u)^2}{2\sigma^2}\right)$$

According to probability density function  $g(x)$ , we know the closer estimation  $f_{(m)}^{\hat{}}$  is to the expectation, the greater the  $g(f_{(m)}^{\hat{}})$ . For ease of calculation, we use Eq. 2 to ignore the effect of  $f_i$  on variance.  $g(f_{(m)}^{\hat{}})$  actually has only one variable—expectation. Separately bring each  $f_{(m)}^{\hat{}}$  into function and multiply these

functions according to maximum likelihood, we get the final target function which needs to be maximized.

$$F(f) = \prod_{m=1}^M g_m(f)$$

We first turn it to logarithmic function  $y = \ln(F(f))$ , and after derivation, the first derivative and the two derivative of  $F(f)$  are obtained sequentially.

$$y' = \frac{\partial \ln(F(f))}{\partial f} = - \sum_{m=1}^M \frac{\hat{f}_{(m)} - f}{\sigma_m^2}$$

$$y'' = \frac{y'}{\partial f} = \sum_{m=1}^M \frac{1}{\sigma_m^2}$$

Through simple analysis,  $y''$  is always bigger than 0 and  $y'$  is a strictly monotone increasing function. So  $F(f)$  is a convex function with a max value. Then set the first derivative function to zero, here when  $\hat{f} = (\sum_{m=1}^M \frac{\hat{f}_{(m)}}{\sigma_m^2}) / (\sum_{m=1}^M \frac{1}{\sigma_m^2})$ , we can get the maximum of the  $F(f)$ .

**B. Proof of Theorem 2**

*Proof.* First use  $t_m$  to denote  $var(\hat{f}_{(m)})$ , the final estimation using maximum likelihood is  $\hat{f} = (\sum_{m=1}^M \frac{\hat{f}_{(m)}}{t_m}) / (\sum_{m=1}^M \frac{1}{t_m})$ . When we calculate the variance of  $\hat{f}$  as follows:

$$var(\hat{f}) = var(\sum_{m=1}^M \frac{\hat{f}_{(m)}}{t_m} / \sum_{m=1}^M \frac{1}{t_m})$$

Since the estimations  $f_m(m \in [M])$  are independent of each other, and  $t_m$  here is actually a constant number.

$$var(\hat{f}) = \sum_{m=1}^M (\frac{var(\hat{f}_m)}{t_m^2}) / (\sum_{m=1}^M \frac{1}{t_m})^2 = 1 / \sum_{m=1}^M \frac{1}{t_m}$$

**C. Proof of Lemma 3**

*Proof.* We still judge the accuracy of the final estimation from the perspective of variance. The Lemma 1 shows base rappor’s estimation variance is  $var(\hat{f}_i) = \frac{e^{\epsilon/2}}{n(e^{\epsilon/2}-1)^2}$ , for the sake of simplicity, let’s first assume  $e^{\epsilon/2} \gg 1$  and use  $t_m$  to denote  $var(\hat{f}_{(m)})$ . So that  $t_m = (1/(n_m e^{\epsilon_m/2}))$ .

We are clear that the  $\hat{f}$ ’s variance and  $\hat{f}_{(m)}$ ’s variance are the same format, because  $f$  is regarded as using Base RAPPOR on the whole population while all users have the same privacy regime  $\epsilon'$ .

Combining the above equations and Theorem 2 together, we can find  $\epsilon' = 2 * \ln \frac{\sum n_m * exp(\epsilon_m)}{\sum n_m}$ . If  $e^{\epsilon/2} \gg 1$  doesn’t hold in some situation, the calculation can still be based on the above formula and the result will become a little more complicated.



## References

1. Akter, M., Hashem, T.: Computing aggregates over numeric data with personalized local differential privacy. In: Pieprzyk, J., Suriadi, S. (eds.) ACISP 2017. LNCS, vol. 10343, pp. 249–260. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-59870-3\\_14](https://doi.org/10.1007/978-3-319-59870-3_14)
2. Apple2017: macos sierra: share analytics information with apple. [https://support.apple.com/kb/PH25654?locale=en\\_US&viewlocale=en\\_US](https://support.apple.com/kb/PH25654?locale=en_US&viewlocale=en_US)
3. Bassily, R., Smith, A.: Local, private, efficient protocols for succinct histograms. In: Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing, pp. 127–135. ACM (2015)
4. Bassily, R., Stemmer, U., Thakurta, A.G., et al.: Practical locally private heavy hitters. In: Advances in Neural Information Processing Systems, pp. 2285–2293 (2017)
5. Chen, R., Li, H., Qin, A.K., Kasiviswanathan, S.P., Jin, H.: Private spatial data aggregation in the local setting. In: IEEE International Conference on Data Engineering, pp. 289–300 (2016)
6. Dwork, C.: Differential privacy. In: International Colloquium on Automata, Languages, and Programming, pp. 1–12 (2006)
7. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006). [https://doi.org/10.1007/11681878\\_14](https://doi.org/10.1007/11681878_14)
8. Dwork, C., Roth, A.: The Algorithmic Foundations of Differential Privacy. Now Publishers Inc., Hanover (2014)
9. Erlingsson, Ú., Korolova, A., Pihur, V.: RAPPOR: randomized aggregatable privacy-preserving ordinal response. In: ACM SIGSAC Conference on Computer and Communications Security, pp. 1054–1067 (2014)
10. Jorgensen, Z., Yu, T., Cormode, G.: Conservative or liberal? Personalized differential privacy. In: 2015 IEEE 31st International Conference on Data Engineering (ICDE), pp. 1023–1034. IEEE (2015)
11. Kairouz, P., Bonawitz, K., Ramage, D.: Discrete distribution estimation under local privacy. arXiv preprint [arXiv:1602.07387](https://arxiv.org/abs/1602.07387) (2016)
12. Kairouz, P., Oh, S., Viswanath, P.: Extremal mechanisms for local differential privacy. In: Advances in Neural Information Processing Systems, pp. 2879–2887 (2014)
13. Kasiviswanathan, S.P., Lee, H.K., Nissim, K., Raskhodnikova, S.: What can we learn privately? In: Proceedings IEEE Annual IEEE Symposium on Foundations of Computer Science, vol. 40, no. 3, pp. 793–826 (2008)
14. Li, H., Xiong, L., Ji, Z., Jiang, X.: Partitioning-based mechanisms under personalized differential privacy. In: Kim, J., Shim, K., Cao, L., Lee, J.-G., Lin, X., Moon, Y.-S. (eds.) PAKDD 2017. LNCS (LNAI), vol. 10234, pp. 615–627. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-57454-7\\_48](https://doi.org/10.1007/978-3-319-57454-7_48)
15. Narayanan, A., Shmatikov, V.: How to break anonymity of the Netflix prize dataset. *Comput. Sci.* (2007)
16. Qin, Z., Yang, Y., Yu, T., Khalil, I., Xiao, X., Ren, K.: Heavy hitter estimation over set-valued data with local differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 192–203. ACM (2016)
17. Tang, J., Korolova, A., Bai, X., Wang, X., Wang, X.: Privacy loss in Apple’s implementation of differential privacy on macOS 10.12. arXiv preprint [arXiv:1709.02753](https://arxiv.org/abs/1709.02753) (2017)

18. Wang, T., Blocki, J., Li, N., Jha, S.: Locally differentially private protocols for frequency estimation. In: Proceedings of the 26th USENIX Security Symposium, pp. 729–745 (2017)
19. Wang, T., Li, N., Jha, S.: Locally differentially private heavy hitter identification. arXiv preprint [arXiv:1708.06674](https://arxiv.org/abs/1708.06674) (2017)
20. Wang, T., Li, N., Jha, S.: Locally differentially private frequent itemset mining. In: IEEE Symposium on Security and Privacy, p. 0. IEEE (2018)
21. Warner, S.L.: Randomized response: a survey technique for eliminating evasive answer bias. *J. Am. Stat. Assoc.* **60**(309), 63–69 (1965)
22. Ye, M., Barg, A.: Optimal schemes for discrete distribution estimation under local differential privacy. In: 2017 IEEE International Symposium on Information Theory (ISIT), pp. 759–763. IEEE (2017)



# Privacy Preserving Elastic Stream Processing with Clouds Using Homomorphic Encryption

Arosha Rodrigo<sup>1</sup>, Miyuru Dayarathna<sup>2</sup>(✉), and Sanath Jayasena<sup>1</sup>

<sup>1</sup> Department of Computer Science and Engineering, University of Moratuwa, Moratuwa, Sri Lanka

uom.arosha@gmail.com, sanath@cse.mrt.ac.lk

<sup>2</sup> WSO2, Inc., Mountain View, USA

miyurud@wso2.com

**Abstract.** Prevalence of the Infrastructure as a Service (IaaS) clouds has enabled organizations to elastically scale their stream processing applications to public clouds. However, current approaches for elastic stream processing do not consider the potential security vulnerabilities in cloud environments. In this paper we describe the design and implementation of an Elastic Switching Mechanism for data stream processing which is based on Homomorphic Encryption (HomoESM). The HomoESM not only does elastically scale data stream processing applications into public clouds but also preserves the privacy of such applications. Using a real world test setup, which includes an email filter benchmark and a web server access log processor benchmark (EDGAR) we demonstrate the effectiveness of our approach. Multiple experiments on Amazon EC2 indicate that the proposed approach for Homomorphic encryption provides significant results which is 10% to 17% improvement of average latency in the case of email filter benchmark and EDGAR benchmarks respectively. Furthermore, EDGAR add/subtract operations and comparison operations showed 6.13% and 26.17% average latency improvements respectively. These promising results pave the way for real world deployments of privacy preserving elastic stream processing in the cloud.

**Keywords:** Cloud computing · Elastic data stream processing · Compressed event processing · Data compression · IaaS · System sizing and capacity planning

## 1 Introduction

Data stream processing conducts online analytics processing on data streams [5]. Data stream processing has applications in diverse areas such as health informatics [1], transportation [16], telecommunications [24], etc. These applications have been implemented on data stream processing engines [5]. Most of the initial data

stream processors were run on isolated computers/clusters (i.e., private clouds). The rise of cloud computing era has resulted in the ability of on demand provisioning of hardware and software resources. This has resulted in data stream processors which run as managed cloud services (e.g., [10, 14]) as well as hybrid cloud services (e.g., Striim [23]).

Stream processing systems often face resource limitations during their operation due to unexpected loads [2, 6]. Several approaches exist which could solve such an issue. Elastically scaling into an external cluster [15, 21], load shedding, approximate query processing [20], etc. are some examples. Out of these, elastic scaling has become a key choice because approaches such as load shedding, approximate computing has to compromise accuracy which is not accepted by certain categories of applications. Previous work has been there which used data compression techniques to optimize the network connection between private and public clouds [21]. However, current elastic scaling mechanisms for stream processing do not consider a very important problem: preserving the privacy of the data sent to public cloud.

Preserving the privacy of stream processing operation becomes one of the key questions to be answered when scaling into a public cloud. Sending the data unencrypted to the server definitely exposes them to prying eyes of the eavesdroppers. Sending data encrypted over the network and decrypting them to get original values at the server may also expose sensitive information. Multiple work has recently being conducted on privacy preserving data stream mining. Privacy of patient health information has been a serious issue in recent times [19]. Fully Homomorphic Encryption (FHE) has been introduced as a solution [9]. FHE is an advanced encryption technique that allows data to be stored and processed in encrypted form. This gives cloud service providers the opportunity for hosting and processing data without even knowing what the data is. However, current FHE techniques are computationally expensive needing excessive space for keys and cypher texts. However, it has been shown with some experiments done with HELib [12] (an FHE library) that it is practical to implement some basic applications such as streaming sensor data to the cloud and comparing the values to a threshold.

In this paper we discuss elastic scaling in a private/public cloud (i.e., hybrid cloud) scenario with privacy preserving data stream processing. We design and implement a privacy preserving Elastic Switching Mechanism (HomoESM) over private/public cloud system. Homomorphic encryption scheme of HELib has been used on top of this switching mechanism for compressing the data sent from private cloud to public cloud. Application logic at the private cloud is implemented with Siddhi event processing engine [16]. We designed and developed two real world data stream processing benchmarks called EmailProcessor and HTTP Log Processor (EDGAR benchmark) during the evaluation of the proposed approach. Using multiple experiments on real-world system setup with the stream processing benchmarks we demonstrate the effectiveness of our approach for elastic switching-based privacy preserving stream processing. We observe that Homomorphic encryption provides significant results which is 10% to 17%

improvement of average latency in the case of Email Filter benchmark. EDGAR comparison and add/subtract operations showed 26.17% average latency improvement. HomoESM is the first known data stream processor which does privacy preserving data stream processing in hybrid cloud scenarios effectively. We have released HomoESM and the benchmark codes as open source software<sup>123</sup>. Specifically, the contributions of our work can be listed as follows.

- *Privacy Preserving Elastic Switching Mechanism (HomoESM)* - We design and develop a mechanism for conducting elastic scaling of stream processing queries over private/public cloud in a privacy preserving manner.
- *Benchmarks* - We design and develop two benchmarks for evaluating the performance of HomoESM.
- *Optimization of Homomorphic Operations* - We optimized several homomorphic evaluation schemes such as equality, less than/greater than comparison. We also do data batching based optimizations.
- *Evaluation* - We evaluate the proposed approaches by implementing them on real world systems.

The paper is organized as follows. Next, we provide related work in Sect. 2. We provide the details of system design in Sect. 3 and implementation of the HomoESM in Sect. 4. The evaluation details are provided in Sect. 5. We make a discussion of the results in Sect. 6. We provide the conclusions in Sect. 7.

## 2 Related Work

There have been multiple previous work on elastic scaling of event processing systems in cloud environments.

Cloud computing allows for realizing an elastic stream computing service, by dynamically adjusting used resources to the current conditions. Hummer *et al.* discussed how elastic computing of data streams can be achieved on top of Cloud computing [13]. They mentioned that the most obvious form of elasticity is to scale with the input data rate and the complexity of operations (acquiring new resources when needed and releasing resources when possible). However, most operators in stream computing are stateful and cannot be easily split up or migrated (e.g., window queries need to store the past sequence of events). In HomoESM we handle this type of queries by query switching.

Stormy is a system developed to evaluate the “stream processing as service” concept [18]. The idea was to build a distributed stream processing service using techniques used in cloud data storage systems. Stormy is built with scalability, elasticity and multi-tenancy in mind to fit in the cloud environment. They have used distributed hash tables (DHT) to build their solution. They have used DHTs to distribute the queries among multiple nodes and to route events from

<sup>1</sup> <https://github.com/aroشارodrigo/event-publisher>.

<sup>2</sup> <https://github.com/aroشارodrigo/statistics-collector>.

<sup>3</sup> <https://github.com/aroشارodrigo/simple-siddhi-server>.

one query to another. Stormy builds a public streaming service where users can add new streams on demand. One of the main limitations in Stormy is it assumes that a query can be completely executed on one node. Hence, Stormy is unable to deal with streams for which the incoming event rate exceeds the capacity of a node. This is an issue which we address in our work via the concept of data switching of HomoESM.

Cervino *et al.* try to solve the problem of providing a resource provisioning mechanism to overcome inherent deficiencies of cloud infrastructure [2]. They have conducted some experiments on Amazon EC2 to investigate the problems that might affect badly on a stream processing system. They have come up with an algorithm to scale up/down the number of VMs (or EC2 instances) based solely on the input stream rate. The goal is to keep the system with a given latency and throughput for varying loads by adaptively provisioning VMs for streaming system to scale up/down. However, none of the above-mentioned works have investigated on reducing the amount of data sent to public clouds in such elastic scheduling scenarios. In this work we address this issue.

Data stream compression has been studied in the field of data mining. Cuzzocrea *et al.* have conducted research on a lossy compression method for efficient OLAP [3] over data streams. Their compression method exploits semantics of the reference application and drives the compression process by means of the “degree of interestingness”. The goal of this work was to develop a methodology and required data structures to enable summarization of the incoming data stream. However, the proposed methodology trades off accuracy and precision for the reduced size.

Dai *et al.* have implemented homomorphic encryption library [4] on Graphic Processing Unit (GPU) to accelerate computations in homomorphic level. As GPUs are more compute-intensive, they show 51 times speedup on homomorphic sorting algorithm when compared to the previous implementation. Although computation wise it gives better speed up, when encrypting a Java String field, its length goes more than 400 KB which is too large to be sent over a public network. Hence we used HELib as the homomorphic encryption library in our work.

Intel has included a special module in CPU, named *Software Guard eXtension (SGX)*, with its 6th generation Core i5, i7, and Xeon processors [22]. SGX reduces the trusted computing base (TCB) to a minimal set of trusted code (programmed by the programmer) and the SGX processor. Shaon *et al.* developed a generic framework for secure data analytics in an untrusted cloud setup with both single user and multi-user settings [22]. Furthermore, they proposed BigMatrix which is an abstraction for handling large matrix operations in a data oblivious manner to support vectorizations. Their work is tailored for data analytics tasks using vectorized computations, and optimal matrix based operations. However, in this work HomoESM conducts stream processing which is different from the batch processing done by BigMatrix.

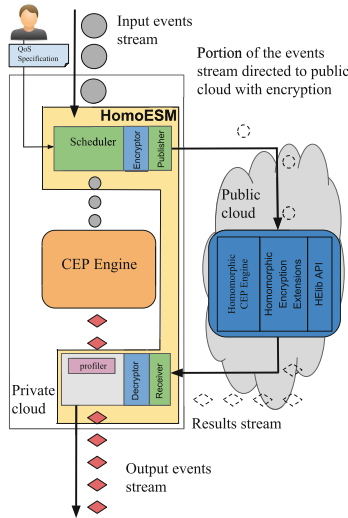
### 3 System Design

In this section we first describe the architecture of HomoESM and then describe the switching functions which determine when to start sending data to public cloud.

The HomoESM architecture is shown in Fig. 1. The components highlighted in the dark blue color correspond to components which directly implement privacy preserving stream processing functionality.

In this system architecture Scheduler collects events from the Plain Event Queue according to the configured frequency and the timestamp field on the event. Then it routes the events into the private publishing thread pool and to the public publishing queue, according to the load transfer percentage and the threshold values.

Receiver receives events from both private & public Siddhi. If the event is from the private Siddhi, it is sent to the Profiler. If not the event is a composite event and it is directed to the ‘Composite Event Decode Worker’ threads located inside the Decryptor which basically performs the decryption function. Finally, all the streams which goes out from HomoESM run through Profiler which conducts the latency measurements.



**Fig. 1.** The system architecture of Homomorphic Encryption based ESM (HomoESM) (Color figure online).

In this paper we use the same switching functions described in [21] for triggering and stopping data sending to public cloud (See Eq. 1). It should be noted that the main contribution of this paper is to describe the elastic privacy preserving stream functionality. Here  $\phi_{VM}(t)$  is the binary switching function for a single VM,  $t$  is the time period of interest.  $L_{t-1}$  and  $D_{t-1}$  are the latency and

data rate values measured in the previous time period. A time period of  $\tau$  has to be elapsed in order for the VM startup process to trigger.  $D_s$  is the threshold for total amount of data received by the VM from private cloud.

$$\phi_{VM}(t) = \begin{cases} 1, L_{t-1} \geq L_s, \tau \text{ has elapsed.} \\ 0, D_{t-1} < D_s, L_{t-1} < L_p & \text{Otherwise} \end{cases} \quad (1)$$

## 4 Implementation

In this Section first we describe the implementation details of HomoESM in Sect. 4.1 and we describe the benchmark implementations in Sects. 4.2, 4.3, 4.4, and 4.5.

### 4.1 Implementation of HomoESM

We have developed the HomoESM on top of the WSO2 Stream Processor (WSO2 SP) software stack. WSO2 SP is an open source, lightweight, easy-to-use, stream processing engine [26]. WSO2 SP internally uses Siddhi which is a complex event processing library [16]. Siddhi feature of WSO2 SP lets users run queries using an SQL like query language in order to get notifications on interesting real-time events.

High-level view of the system implementation is shown in Fig. 2. Input events are received by the ‘Event Publisher’. Java objects are created for each incoming event and put into a queue. Event publisher thread picks those Java objects from the queue according to the configured period. Next, it evaluates whether the picked event needs to be sent to the private or the public Siddhi server, according to the configured load transfer percentage and threshold values. If that event needs to be sent to private Siddhi, it will mark the time and delegate the event into a thread pool which handles sending to private Siddhi. If that event needs to be sent to public Siddhi, it will mark the time and put into the queue which is processed by the Encrypt Master asynchronously.

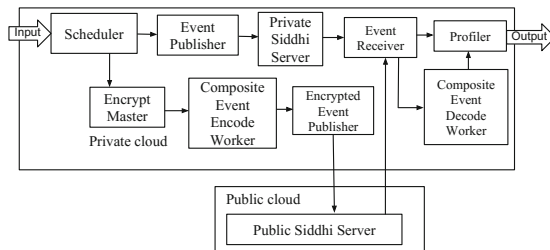


Fig. 2. Main components of HomoESM

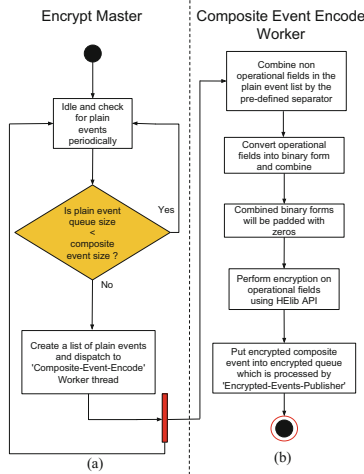


Encrypt Master thread (see Fig. 3(a)) periodically checks a queue which keeps the events required to be sent to public cloud. The queue is maintained by the ‘Event Publisher’ (See Fig. 4(a)). If that queue size is greater than or equal to composite event size, it will create a list of events equal to the size of composite event size. Next, it delegates the event encryption and composite event creation task to the ‘Composite Event Encode Worker’ (see Fig. 3(b)).

Composite Event Encode Worker is a thread pool which handles event encryptions and composite event creations. First, it combines non-operational fields of each plain events in the list by the pre-defined separator. Then it converts operational fields into binary form and combines them together. Next, it pads the operational fields with zeros, in order to encrypt using HELIB API. Finally, it performs encryption on those operational fields and puts the newly created composite event into a queue which is processed by the ‘Encrypted Events Publisher’ thread (See Fig. 4(b)).

Firing events into the public VM is done asynchronously. Decision of how many events sent to the public Siddhi server was taken according to the percentage we have configured initially. But the public Siddhi server’s publishing flow has max limit of 1500 TPS (Tuples Per Second). If the Event Publisher receives more than the max TPS, the events are routed back into the private Siddhi server’s VM.

‘Encrypted Events Publisher’ thread periodically checks for encrypted events in the encrypted queue which is put by the ‘Composite Event Encode Worker’ at the end of the composite event creation and encryption process (See Fig. 3(b)). First, it combines non-operational fields of each plain event in the list by the pre-defined separator. If there are encrypted events, it will pick those at once and send to public Siddhi server. The Encryptor module batches events into



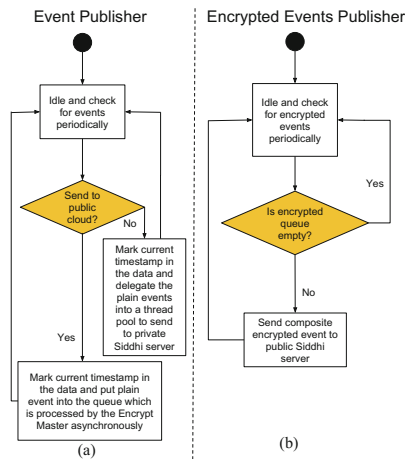
**Fig. 3.** Data encryption and the composite event creation process at the private Siddhi server. (a) Encrypt Master thread (b) Composite Event Encode Worker thread

composite events and encrypts each composite message using Homomorphic encryption. The encrypted events are sent to the public cloud where Homomorphic CEP Engine module conducts the evaluation.

We encrypt operand(s) and come up with composite operand field(s) in each HE function initially, in order to perform HE operations on operational fields in composite event. For example, in the case of the Email Filter benchmark, at the Homomorphic CEP engine which supports Homomorphic evaluations, initially it converts the constant operand into an integer (int) buffer with size 40 with a necessary 0 padding. Then it replicates the integer buffer 10 times and encrypts using HELib [11]. Finally, the encrypted value and the relevant field in the composite event are used for HELib’s relevant (e.g., comparison, addition, subtraction, division, etc.) operation homomorphically. The result is replaced with the relevant field in the composite event and is sent to the Receiver without any decryption.

The received encrypted information is decrypted and decomposed to extract the relevant plain events. The latency measurement happens at the end of this flow. ‘Event Receiver’ thread checks if the event received from the Siddhi server is encrypted with Homomorphic encryption. If so it delegates composite event into ‘Composite Event Decode Worker’. If not it will read payload data and calculate the latency (See Fig. 5(a)).

After receiving a composite event from the Event Receiver the Composite Event Decode Worker handles all decomposition and decryptions of the composite event (See Fig. 5(b)). It first splits non-operational fields in the composite event by the pre-defined separator. Second, it performs decryption on the operational fields using HELib API and splits the decrypted fields into fixed-length strings. Then it creates plain events using the splitted fields. Next, it checks each



**Fig. 4.** Operation of the Event Publisher and the Encrypted Events Publisher components. (a) Event Publisher (b) Encrypted Events Publisher

operational fields in the plain event to see whether it contains zeros and then processes the events. Finally, it calculates the latency of the decoded events.

Note that we implement the Homomorphic comparison of values following the work by Togan *et al.* [25]. For two single bit numbers with  $x$  and  $y$ , Togan *et al.* [25] have shown that the following equations (see Eq. 2) will satisfy greater-than and equal operations, respectively.

$$\begin{aligned}
 x > y &\Leftrightarrow xy + x = 1 \\
 x = y &\Leftrightarrow x + y + 1 = 1
 \end{aligned}
 \tag{2}$$

Togan *et al.* have created comparison functions for  $n$ -bit numbers using divide and conquer methodology. In our case we derived 2-bit number comparisons as follows.  $x_1x_0$  and  $y_1y_0$  are the two numbers with 2-bits (see Eq. 3). Here every ‘+’ operation is for XOR gate operation and every ‘.’ operator is for AND gate operation.

$$\begin{aligned}
 x_1x_0 > y_1y_0 &\Leftrightarrow (x_1 > y_1) \vee (x_1 = y_1) \wedge (x_0 > y_0) = 1 \\
 &\Leftrightarrow (x_1.y_1 + x_1) + (x_1 + y_1 + 1)(x_0.y_0 + x_0) = 1 \\
 &\Leftrightarrow x_1.y_1 + x_1 + x_1.x_0.y_0 + x_1.x_0 + \\
 &\quad y_1.x_0.y_0 + y_1.x_0 + x_0.y_0 + x_0 = 1 \\
 x_1x_0 = y_1y_0 &\Leftrightarrow (x_0 + y_0 + 1).(x_1 + y_1 + 1) = 1 \\
 &\Leftrightarrow x_0.x_1 + x_0.y_1 + x_0 + y_0.x_1 + y_0.y_1 + y_0 + 1 = 1 \\
 x_1x_0 < y_1y_0 &\Leftrightarrow (x_1x_0 > y_1y_0) + (x_1x_0 = y_1y_0) + 1 = 1 \\
 &\Leftrightarrow (x_1.y_1 + x_1 + x_1.x_0.y_0 + x_1.x_0 + y_1.x_0.y_0 \\
 &\quad + y_1.x_0 + x_0.y_0 + x_0) + (x_0.x_1 + x_0.y_1 \\
 &\quad + x_0 + y_0.x_1 + y_0.y_1 + y_0 + 1) + 1 = 1
 \end{aligned}
 \tag{3}$$

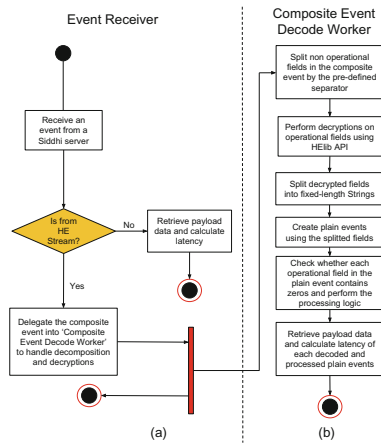


Fig. 5. Event receiving, decomposition, and decryption processes.

Reason that we build up comparison functions for two bit numbers is to apply the concept of homomorphic encryption and evaluation into the CEP engine. Even for 2-bit number comparisons, there are a number of XOR and AND gate evaluations need to be done as above.

After evaluating the individual HE operations at public SP, filtering using those gate operations happens at private SP. Boolean conditions are evaluated on encrypted operands using HE with above limitations for input number range, and ‘NOT’, ‘AND’, and ‘OR’ gate operations evaluate at private SP after decrypting/decoding the events which comes from public SP after HE evaluations.

We have evaluated the HomoESM’s functionality using four benchmark applications developed using two data sets. Next, in order to ensure the completeness of this section we describe the implementation details of the two benchmarks.

### 4.2 Email Filter Benchmark

Email Filter is a benchmark we developed based on the canonical Enron email data set [17]. The data set has 517,417 emails with an average body size of 1.8KB, the largest being 1.92 MB. The Email Filter benchmark only had filter operation and was used to compare filtering performance compared to the EDGAR Filter benchmark which is described in the next subsection. The architecture of the Email Filter benchmark is shown in Fig. 6. The events in the input emails stream had eight fields *ij\_timestamp*, *fromAddress*, *toAddresses*, *ccAddresses*, *bccAddresses*, *subject*, *body*, *regexstr* where all the fields were Strings except *ij\_timestamp* which was long type. We formatted the *toAddresses* and *ccAddresses* fields to have only single email address to support HELib evaluations. The criteria for filtering out Emails was to filter by the email addresses *lynn.blair@enron.com* and *richard.hanagriff@enron.com*. The filtering SiddhiQL statement can be stated as in Listing 1.1,

```

NOT ((fromAddress is equal to ‘lynn.blair@enron.com’) AND
((toAddresses is equal to ‘richard.hanagriff@enron.com’)
OR (ccAddresses is equal to ‘richard.hanagriff@enron.com’)
)))
    
```

Listing 1.1. EmailFilter condition.

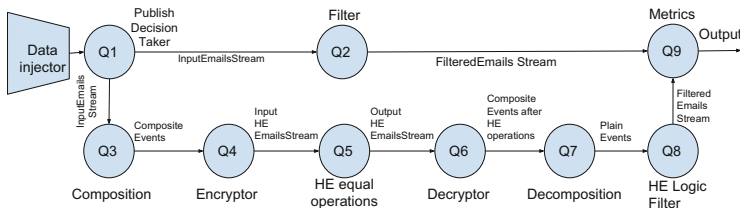


Fig. 6. Architecture of Email Filter benchmark.

### 4.3 EDGAR Filter Benchmark

We developed another benchmark based on a HTTP log data set published by Division of Economic and Risk Analysis (DERA) [8]. The data provides details of the usage of publicly accessible EDGAR company filings in a simple but extensive manner [8]. Each record in the data set consists of 16 different fields hence each event sent to the benchmark had 16 fields (*ij\_timestamp*, *ip*, *date*, *time*, *zone*, *cik*, *accession*, *extension*, *code*, *size*, *idx*, *norefer*, *noagent*, *find*, *crawler*, and *browser*). Similar to the Email Filter benchmark all of the fields except *ij\_timestamp* were Strings. Out of these fields we used *noagent* field by adding lengthy string of 1024 characters to the existing value, in order to increase the events' size (Note that we have done the same for all the EDGAR benchmarks described in this paper).

The EDGAR benchmark was developed with the aim of implementing filtering support. Basic criteria was to filter out EDGAR logs, which satisfy the conditions shown in Listing 1.2.

```
(extension == 'v16003sv1.htm') and (code ==
'200.0') and (date == '2016-10-01'))
```

**Listing 1.2.** EDGAR filter condition.

Most of the EDGAR log events were same and the logs did not have any data rate variation inherently. Therefore, we introduced varying data rate by publishing events in different TPS values according to a custom-defined function.

### 4.4 EDGAR Comparison Benchmark

Using the same EDGAR data set we developed EDGAR Comparison benchmark to evaluate the performance [7] of Homomorphic Comparison operation. In the EDAGR Comparison benchmark We have changed the input format of the *zone* and *find* fields to integer (Int) in order to do comparison operations. Since we are doing only bitwise operations, we limited the HELib message space to 2, in order to use only 0s and 1s. Therefore, maximum length for encrypting field when we used message space as 2 was 168, and we used composite event size as 168 when sending to public Siddhi server. The architecture of EDGAR Comparison benchmark is similar to the topology shown in Fig. 6. Basic criteria is to filter out EDGAR logs, which satisfy following conditions (See Listing 1.3).

```
(zone == 0) and (find > 0) and (find < 3)
```

**Listing 1.3.** EDGAR comparison condition.

### 4.5 EDGAR Add/Subtract Benchmark

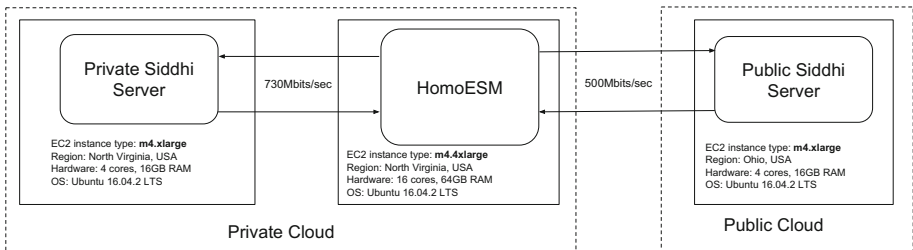
In EDGAR add/subtract benchmark we have changed the input format to an Integer, for *code*, *idx*, *norefer*, and *find* fields in order to support add/subtract operations. The corresponding siddhi query which depicts the addition and subtract operations conducted by this benchmark is shown in Listing 1.4.

```
@info(name = 'query5') from
inputEdgarStream select iij_timestamp, ip, date, time,
zone, cik, accession, extension, code-100 as code, size,
idx+30 as idx, norefer+20 as norefer, noagent, find-10 as
find, crawler, browser insert into outputEdgarStream;
```

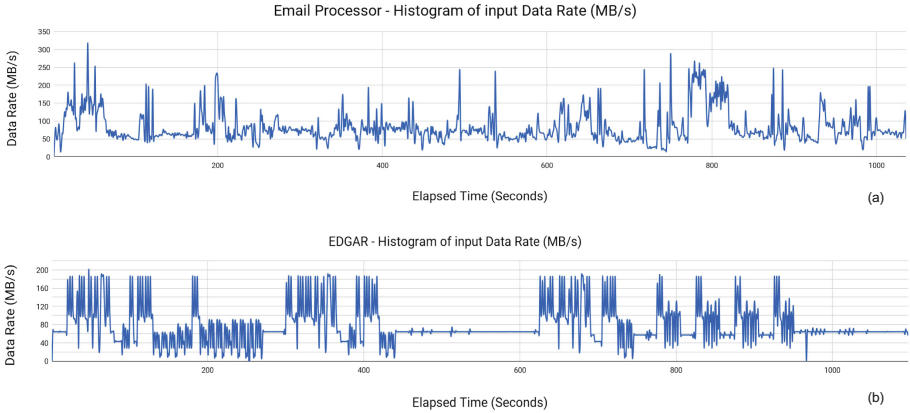
**Listing 1.4.** EDGAR add/subtract siddhi query.

## 5 Evaluation

We conducted the experiments using three VMs in Amazon EC2. In this experiment two VMs were hosted in North Virginia, USA and they were used as private cloud while the VM used as public cloud was located in Ohio, USA. We used the Email Filter benchmark in this experiment which does filtering of an email event stream. Out of the two VMs in North Virginia one was a *m4.xlarge* instance which had 16 cores, 64 GB RAM while the private CEP Engine was deployed in a *m4.xlarge* instance which had 4 CPU cores, 16 GB RAM. In *m4.xlarge* VM we have deployed ‘event-publisher’ (Event Publisher) and ‘statistic-collector’ (Event Receiver) modules. The Stream Processor engine running in the public cloud was deployed on the VM running in Ohio which was a *m4.xlarge* instance. All the VMs were running on Ubuntu 16.04.2 LTS (Long Term Support). Using a network speed measurement tool we observed that network speed between the two VMs in North Virginia was around 730 Mbits/s while the network speed between North Virginia and Ohio was 500 Mbits/s. Figure 7 shows the architecture of the experiment setup. The input data rate variation of the Email benchmark and the EDGAR benchmark data sets is shown in Fig. 8(a) and (b) respectively. The two charts indicate that the workloads imposed by the two benchmarks have significantly different characteristics.



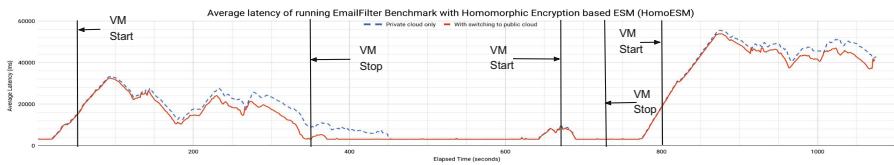
**Fig. 7.** Experiment setup of HomoESM on Amazon EC2.



**Fig. 8.** Input data rate variation of the two benchmarks (a) Email Filter benchmark (b) EDGAR benchmarks.

### 5.1 Email Filter Benchmark

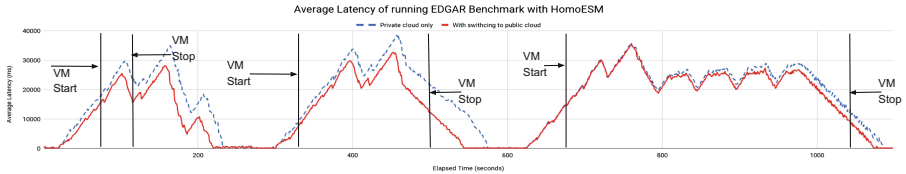
In the first round we used Email Filter benchmark. The results of this experiment is shown in Fig. 9. The curve in the blue color (dashed line) indicates the private cloud deployment. The red color curve indicates the deployment with switching to public cloud. It can be observed a clear reduction of average latency when switched to the public cloud in this setup compared to the private cloud only deployment. With homomorphic elastic scaling an overall average latency reduction of 2.14 seconds per event can be observed. This is 10.24% improvement compared to the private cloud only deployment. Note that in all the following charts we have marked the times where VM start/VM stop operations have been invoked in order to start/stop the VM in the public cloud. Since VM startup and data sending times are almost similar, in this paper we assume VM startup time as the data sending time and VM stop time as the point where we stop sending data to public cloud.



**Fig. 9.** Average latency of elastic scaling of the Email Filter benchmark with securing the event stream sent to public cloud via homomorphic encryption (Color figure online).

### 5.2 EDGAR Filter Benchmark

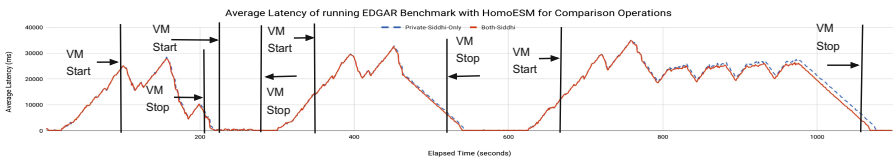
In the second round we used EDGAR Filter benchmark for evaluation of our technique. The results are shown in Fig. 10. It can be observed significant performance gain in terms of latency when switching to public cloud with the EDGAR benchmark. A notable fact is that EDGAR data set had relatively smaller message size. The average message size of the EDGAR benchmark was 1.1 KB. The HomoESM mechanism was able to reduce the delay with considerable improvement of 17%.



**Fig. 10.** Average latency of elastic scaling of the EDGAR benchmark with Homomorphic filter operations.

### 5.3 EDGAR Comparison Benchmark

Next, we evaluated the Homomorphic comparison operation. Here we have used a slightly modified version of the EDGAR Filter benchmark to facilitate comparison operation in a homomorphic manner. Here also we add lengthy string of 1024 characters to the existing value of ‘noagent’ field. The results are shown in Fig. 11.



**Fig. 11.** Average latency of elastic scaling of the EDGAR benchmark with Homomorphic comparison operations.

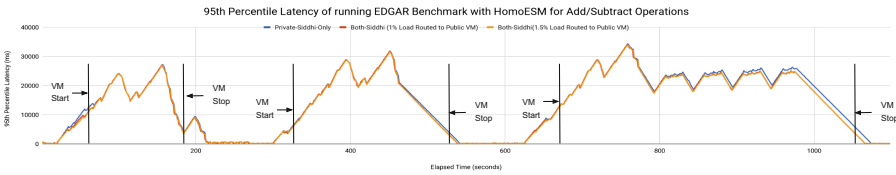
We could see only a slight improvement of latency with EDGAR comparison benchmark. The improvement of the average latency was around 449 ms which is 3% improvement compared to the private only deployment. Compared to equal only operation, less-than & greater-than operations consume more XOR & AND gate operations in the Homomorphic Encryption (HE) level. Due to that Siddhi engine processing throughput, when having homomorphic less-than & greater-than operations is quite low compared to equal operation only case. Therefore, the portion of events sent to public Siddhi is lesser than other cases. That’s why



we could not see much advantage (only 3%) on latency curves for both private & public Siddhi setup compared to private Siddhi only setup. During the middle spike shown in Fig. 11, a 26.17% improvement in latency was observed.

#### 5.4 EDGAR Add/Subtract Benchmark

Finally, we evaluated the Homomorphic add/subtract operation using the EDGAR benchmark. The addition and subtraction HE operations' supported message space range is from 0 to 1201. Although 32-bit full adder circuits using HELib could increase the range further we keep this as a further work. The overall improvement was 3.68% for the scenario where 1.5% of the load was sent to the Public VM. We observed a maximum 6.13% performance improvement in the third spike shown in Fig. 12.



**Fig. 12.** Average latency of elastic scaling of the EDGAR benchmark with Homomorphic Add/Subtract operations.

## 6 Discussion

Privacy preserving data mining in clouds has been an area of significant interest in recent times. However, none of the previous work on elastic stream processing has demonstrated the feasibility of conducting elastic privacy preserving data stream processing. In this paper we have not only implemented a mechanism for elastic privacy preserving data stream processing but also have shown considerable performance benefits on real world experiment setups. Results comparing HomoESM to the private cloud only deployments demonstrate 3–17% latency improvements. Furthermore, during large workload spikes HomoESM has shown 6–26% latency improvements which is almost doubled performance improvement. Workload spikes are the key situations where HomoESM needs to be deployed which indicates HomoESM's effectiveness in handling such situations.

Although one could argue that the techniques presented in this paper are restricted due to the nature of the modern homomorphic encryption techniques, we have overcome the difficulties via batching and compressing the events, which is one of the key contributions of this paper. We have used high performance VM instance type m4.4xlarge in the evaluations, because composite event composing & decomposing require more CPU for publisher and statistics collector. A limitation of FHE is that it needs prior knowledge of the data to conduct different operations on the encrypted data. Hence, HomoESM is applicable only for data streams with finite and unchanging data.

## 7 Conclusion

Privacy has become an utmost important barrier which hinders leveraging IaaS for running stream processing applications. In this paper we introduce a mechanism called HomoESM which conducts privacy preserving elastic data stream processing. We evaluated our approach using two benchmarks called Email Filter and EDGAR on Amazon AWS. We observed significant improvements of overall latency of 10% and 17% for Email Processors and EDGAR data sets with using HomoESM on equality operation. We also implemented comparison and add/subtract operations in HomoESM which resulted in maximum 26.17% and 6.13% improvement in the average latencies respectively. In future, we plan to extend this work to handle more complicated streaming operations. We also plan to experiment with multiple query based tuning for privacy preserving elastic scaling.

## References

1. Blount, M., et al.: Real-time analysis for intensive care: development and deployment of the artemis analytic system. *IEEE Eng. Med. Biol. Mag.* **29**(2), 110–118 (2010)
2. Cervino, J., Kalyvianaki, E., Salvachua, J., Pietzuch, P.: Adaptive provisioning of stream processing systems in the cloud. In: 2012 IEEE 28th International Conference on Data Engineering Workshops (ICDEW), pp. 295–301, April 2012
3. Cuzzocrea, A., Chakravarthy, S.: Event-based lossy compression for effective and efficient OLAP over data streams. *Data Knowl. Eng.* **69**(7), 678–708 (2010)
4. Dai, W., Sunar, B.: cuHE: a homomorphic encryption accelerator library. *Cryptology ePrint Archive, Report 2015/818* (2015). <https://eprint.iacr.org/2015/818>
5. Dayarathna, M., Perera, S.: Recent advancements in event processing. *ACM Comput. Surv.* **51**(2), 33:1–33:36 (2018)
6. Dayarathna, M., Suzumura, T.: A mechanism for stream program performance recovery in resource limited compute clusters. In: Meng, W., Feng, L., Bressan, S., Winiwarter, W., Song, W. (eds.) DASFAA 2013. LNCS, vol. 7826, pp. 164–178. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-37450-0\\_12](https://doi.org/10.1007/978-3-642-37450-0_12)
7. Dayarathna, M., Suzumura, T.: A performance analysis of system S, S4, and esper via two level benchmarking. In: Joshi, K., Siegle, M., Stoelinga, M., D’Argenio, P.R. (eds.) QEST 2013. LNCS, vol. 8054, pp. 225–240. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-40196-1\\_19](https://doi.org/10.1007/978-3-642-40196-1_19)
8. DERA. Edgar log file data set (2017). <https://www.sec.gov/dera/data/edgar-log-file-data-set.html>
9. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, STOC 2009, pp. 169–178. ACM, New York (2009)
10. Google: Cloud dataflow (2017). <https://cloud.google.com/dataflow/>
11. Halevi, S.: An implementation of homomorphic encryption (2017). <https://github.com/sha1h/HElib>
12. Halevi, S., Shoup, V.: Algorithms in HElib. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 554–571. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-44371-2\\_31](https://doi.org/10.1007/978-3-662-44371-2_31)

13. Hummer, W., Satzger, B., Dustdar, S.: Elastic stream processing in the cloud. *Wiley Interdisc. Rev. Data Min. Knowl. Discovery* **3**(5), 333–345 (2013)
14. IBM: Streaming analytics (2017). <https://www.ibm.com/cloud/streaming-analytics>
15. Ishii, A., Suzumura, T.: Elastic stream computing with clouds. In: 2011 IEEE 4th International Conference on Cloud Computing, pp. 195–202, July 2011
16. Jayasekara, S., Perera, S., Dayarathna, M., Suhothayan, S.: Continuous analytics on geospatial data streams with WSO2 complex event processor. In: Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems, DEBS 2015, pp. 277–284. ACM, New York (2015)
17. Klimt, B., Yang, Y.: Introducing the enron corpus, p. 2, January 2004
18. Loesing, S., Hentschel, M., Kraska, T., Kossmann, D.: Stormy: an elastic and highly available streaming service in the cloud. In: Proceedings of the 2012 Joint EDBT/ICDT Workshops, EDBT-ICDT 2012, pp. 55–60. ACM, New York (2012)
19. Page, A., Kocabas, O., Ames, S., Venkitasubramaniam, M., Soyata, T.: Cloud-based secure health monitoring: optimizing fully-homomorphic encryption for streaming algorithms. In: 2014 IEEE Globecom Workshops (GC Wkshps), pp. 48–52, December 2014
20. Quoc, D.L., Chen, R., Bhatotia, P., Fetzer, C., Hilt, V., Strufe, T.: Streamapprox: approximate computing for stream analytics. In: Proceedings of the 18th ACM/I-FIP/USENIX Middleware Conference, Middleware 2017, pp. 185–197. ACM, New York (2017)
21. Ravindra, S., Dayarathna, M., Jayasena, S.: Latency aware elastic switching-based stream processing over compressed data streams. In: Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering, ICPE 2017, pp. 91–102. ACM, New York (2017)
22. Shaon, F., Kantarcioglu, M., Lin, Z., Khan, L.: SGX-BigMatrix: a practical encrypted data analytic framework with trusted processors. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, pp. 1211–1228. ACM, New York (2017)
23. Striim: Striim delivers streaming hybrid cloud integration to microsoft azure (2017). <http://www.striim.com/press/hybrid-cloud-integration-to-microsoft-azure>
24. Theeten, B., Bedini, I., Cogan, P., Sala, A., Cucinotta, T.: Towards the optimization of a parallel streaming engine for telco applications. *Bell Labs Tech. J.* **18**(4), 181–197 (2014)
25. Togan, M., Plesca, C.: Comparison-based computations over fully homomorphic encrypted data. In: 2014 10th International Conference on Communications (COMM), pp. 1–6, May 2014
26. WSO2: WSO2 stream processor (2018). <https://wso2.com/analytics-and-stream-processing>



# Select the Best for Me: Privacy-Preserving Polynomial Evaluation Algorithm over Road Network

Wei Song<sup>(✉)</sup>, Chengliang Shi, Yuan Shen, and Zhiyong Peng

School of Computer Science, Wuhan University, Wuhan, China  
{songwei, chengliangshi, yuanshen, peng}@whu.edu.cn

**Abstract.** Recent years have witnessed the rapid advances in location based services (LBSs) with the fast development of mobile devices and communication technologies. Outsourcing spacial databases to Cloud provides an economical and flexible way for the LBS provider to deploy services. For an LBS enterprise, the collected data might be its most valuable strategic asserts. However, in the outsourced database paradigm, the third-party database server at cloud is not completely trustworthy, therefore, protecting data privacy is critical. A polynomial evaluation algorithm over road network returns top- $k$  results (restaurant) to a user (tourist). It is a fundamental and important query mode has been widely used in LBS applications. In this paper, we extend the Order-Revealing Encryption (ORE) to design a privacy-preserving polynomial evaluation algorithm over the spacial data with security guarantee. To reduce the computation overhead, we introduce an influence model to store the encrypted point data (restaurant). Besides the stronger security, this work is a practical polynomial evaluation algorithm over the road network. The practicality is mainly manifested in two aspects: evaluation over multiple attributes, and the dynamic evaluation function rather than fixed function in advance. We formally prove the security of our scheme in the random oracle model. Finally, we implement a prototype to evaluate the performance of our scheme. The experimental results over the real road network dataset demonstrate that the proposed scheme is an efficient and practical polynomial evaluation algorithm for the LBS applications.

**Keywords:** Order-revealing encryption · Privacy-preserving polynomial evaluation algorithm · Road network ·  $k$ NN query

## 1 Introduction

The development of location-aware smartphone is leading a new wave of location-based services (LBSs). Crowd-sourced review forum is probably the most commonly used of all service models in LBS applications (such as Yelp, TripAdvisor, and Meituan). Using smartphones, any user can rate and review the points of interests (POIs) after he enjoys their services.

By these rating data and reviews, a user is easy to select the best POIs which are near and appropriate for himself, even though he never visited them. Taking Yelp as an example, users (customers) of this application would like to submit their reviews of some POIs (restaurants) using Yelp's rating system after their consumptions. These reviews are collected by Yelp rating system to evaluate all the POIs in a city. While a user attempts to access the POIs around him, the rating scores can help him to make his decision. Receiving the query request with user's location, the LBS server can use the evaluation algorithm to pick up the candidate best surrounding POIs for him.

With the rapid growth of LBS's dataset, the query processes inevitably introduce the huge overheads of computation and storage. Furthermore, the complexity of evaluation algorithm based on the user location depends heavily on the denseness of the POIs around him. It also leads to a high overhead, especially in a big city. To lighten the burden, more and more LBS providers tend to delegate the dataset and query operations to a cloud service provider (CSP), which provides storage and online query services such as Amazon S3.

The data owner, for example, an enterprise which provides LBS, can farm out the dataset to the cloud, and CSP will process the query request according to the previously agreed protocol. There is no doubt that the outsourced service mode provides an efficient and economic way for the enterprise to implement LBS applications, however, it also raises the security concerns. For an LBS enterprise, the collected dataset including the rating scores and the reviews for all the POIs might be its most valuable strategic asserts. There are considerable risks that the malicious attackers will attack the data center for enormous commercial values of these data. Moreover as the CSP is not completely trustworthy, directly outsourcing these data may do harm to the enterprise's interests. Therefore, it is very important for an LBS provider to conduct a privacy-preserving service mechanism in the cloud-based outsourced data management model.

At present, there have been a number of research efforts [3, 4, 9, 10, 17, 18, 20] focused on the privacy-preserving query for the scenario of LBS. Nonetheless, most of them are  $k$ NN queries which adopt the Euclidean distance as the evaluating indicator. Even though some methods provide good performances, the  $k$ NN query based on Euclidean distance still cannot satisfy the requirements of some real-world applications.

For a practically privacy-preserving polynomial evaluation algorithm in LBS applications, we think there are three below challenges that need to be overcome:

- The POIs in a city are distributed along the road. The user can only visit the desired POIs through the paths in the road network. So, a user is more interested in the POIs which are near to him on the road network than those are near in Euclidean distance. How to evaluate the distance in the road network without leaking out privacy is the first challenge.
- Besides the distance factor, a user usually considers the rating scores to select the best POIs. Therefore, a privacy-preserving polynomial evaluation algorithm over multiple attributes (such as distance, rating score) is highly desirable for the real LBS applications.

- The polynomial evaluation algorithm should be personalized, which contains two meanings: the parameters of the polynomial can be decided by the user, and the evaluation polynomial is not fixed in advance.

To address this issue, we extend the Order-Revealing Encryption (ORE) to design a privacy-preserving polynomial evaluation algorithm which meets all the practical requirements mentioned above with strong security guarantee. The main contributions of this paper can be summarized as:

- First, we propose a novel homomorphic and ORE scheme to encrypt the spatial data and the rating scores without leaking out the sensitive information.
- Second, we design a privacy-preserving polynomial evaluation algorithm over road network. By it, the CSP is allowed to directly execute polynomial computation over the encrypted data with strong security guarantee.
- Third, we introduce an influence model to evaluate the influence ranges of the point and the querier to reduce the overheads of storage and computation.
- Finally, we formally prove the security of the proposed scheme in the random oracle model.

## 2 Related Work

The  $k$ NN query is an important query type applied in many fields such as data mining and LBS applications. In order to reserve the distance relationship between the candidate points, Wong et al. [15] proposed an asymmetric scalar-product-preserving encryption (ASPE) scheme as distance-recoverable encryptions. It used an invertible matrix as the encryption key for the database and query request. Some researches have been done on the approximate Privacy Preserving  $k$ NN (PP $k$ NN) query. Yiu et al. [19] proposed three transformation techniques providing some trade-offs among data privacy, query cost and accuracy. Yao et al. [16] proposed a privacy-preserving  $k$ NN method based on partitioned Voronoi diagram. But all the works above cannot execute  $k$ NN query accurately. To provide the accurate PP $k$ NN service, Zhu et al. [22] proposed a query scheme based on Paillier cryptosystem, and Zhou et al. [21] proposed an efficient  $k$ NN query scheme which only reveals limited information of data owner. Both of them are not secure when the cloud server attempts to obtain the data access patterns. In addition, Elmehdwi et al. [2] offered a secure query scheme which allows two non-colluding clouds to perform the PP $k$ NN query.

The PP $k$ NN over road network is much more complicated. In a typical LBS application, not only the POI dataset will cause the leakage of business secret, but also the data access pattern and query requests will reveal the user's privacy. Mix zone [9] can preserve the location privacy of query request, but it is vulnerable under the background knowledge attack. Hu et al. [5] proposed a secure traversal framework which can be categorized as a scheme based on geographic data transformation. Papadopoulos et al. [10] and Yi et al. [17] also proposed secure  $k$ NN query algorithms based on Private Information Retrieval (PIR) schemes [3, 4]. Yi et al. [18] proposed a much more practical scheme which

can be applied to multiple discrete type attributes of private location-based queries. Besides, Paulet et al. [12] provided a method which combines PIR with a stage of oblivious transfer(OT), which maps user’s location into the public grid and maps POIs into the private grid. Liu et al. [7] extended this model with two rounds of oblivious transfer extension. But both of them didn’t provide personalized services. Taking the road networks into consideration, Zhou et al. [20] proposed a practical  $k$ NN query scheme over road networks, but it is not scalable enough for issuing a personalized query request. Song et al. [13] proposed a verifiable scheme for the delegated polynomial functions at the cloud server.

To the best knowledge of us, the practically privacy-preserving polynomial evaluation algorithm over road network is still lacking. It is the main motivation of this work.

### 3 Preliminaries

#### 3.1 Order-Revealing Encryption

An order-revealing encryption (ORE) scheme [1,6] is a tuple of algorithms  $\Pi=(\text{ORE.Setup}, \text{ORE.Encrypt}, \text{ORE.Compare})$ :

- $\text{ORE.Setup}(1^\lambda) \rightarrow sk$ : On input a security parameter  $\lambda$ , the setup algorithm outputs a secret key  $sk$ .
- $\text{ORE.Encrypt}(sk, m) \rightarrow ct$ : On input a secret key  $sk$  and a message  $m \in \mathcal{D}$ , the encryption algorithm outputs a ciphertext  $ct$ .
- $\text{ORE.Compare}(ct_1, ct_2) \rightarrow b$ : On input two ciphertexts  $ct_1, ct_2$ , the compare algorithm outputs a bit  $b \in \{0, 1\}$ .

**Correctness:** We say an ORE scheme defined over a well-ordered domain  $\mathcal{D}$  is correct if for  $sk \leftarrow \text{ORE.Setup}(1^\lambda)$  and all message  $m_1, m_2 \in \mathcal{D}$ ,

$$\Pr[\text{ORE.Compare}(ct_1, ct_2) = 1(m_1 < m_2)] = 1 - \text{negl}(\lambda),$$

in which  $\text{negl}(\lambda)$  denotes a negligible function in  $\lambda$ .

#### 3.2 Homomorphic Encryption

Homomorphic encryption is a form of encryption that allows computation on ciphertexts, generating an encrypted result which, when decrypted, matches the result of the operations as if they had been performed on the plaintext.

In this paper, we adopt Paillier algorithm [11], which supports unlimited number of homomorphic additions between ciphertexts and homomorphic multiplication between a ciphertext and a scalar constant, to implement the privacy-preserving polynomial evaluation algorithm. The Paillier cryptosystem has a plaintext space  $\mathbb{Z}_N$  and a ciphertext space  $\mathbb{Z}_{N^2}$ . Suppose  $E(\cdot)$  and  $D(\cdot)$  are the encrypting operator and decrypting operator. For a message  $m \in \mathbb{Z}_N$ , the Paillier encryption is given as  $E(m) = g^m r^N \bmod N^2$ , where  $g$  is the public key and  $r$  is a random number. For a ciphertext  $ct$ , the Paillier decryption is given as

$D(ct) = L(ct \bmod N^2) / L(g^\lambda \bmod N^2) \bmod N$ , where  $\lambda$  is the decryption key. The Paillier cryptosystem has additive homomorphic property given as

$$D(E(m_1) \cdot E(m_2)) = m_1 + m_2 \bmod N^2$$

$$D(E(m)^a) = a \cdot m \bmod N^2, \forall a \in \mathbb{Z}.$$

**Definition 1.** *In this paper, we build the privacy-preserving homomorphic additive encryption algorithm CMP based on Paillier cryptosystem as*

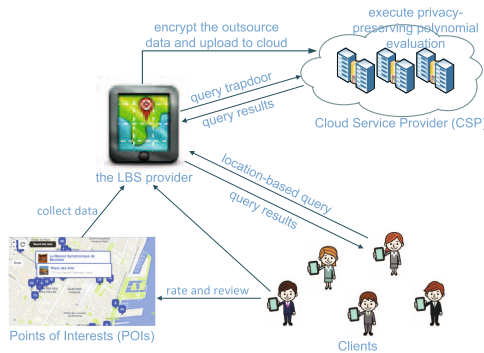
$$CMP(m_1, m_2) = E(m_1) \cdot E(m_2). \tag{1}$$

The encryption algorithm CMP has the following properties: (1)  $CMP(m_1, m_2) = E(m_1 + m_2)$ ; (2)  $CMP(m_1, m_2) \cdot CMP(m_3, m_4) = E(m_1 + m_2 + m_3 + m_4)$ ; (3)  $CMP(m_1, m_2)^a = E(am_1 + am_2) = CMP(am_1, am_2)$ .

## 4 Problem Statement

### 4.1 System Model

In this work, we consider an LBS system composed of three entities as shown in Fig. 1, i.e., the cloud service provider (CSP), the LBS provider, and the clients. The LBS provider collects the data of points (restaurant, hotel and so on) in a city and provides the location-based recommendation services to the clients. The collected data include the points' locations and the rating scores. As the huge scale of the collected data, the LBS provider uploads the data and delegates the evaluation computation to the CSP which has strong computation power.



**Fig. 1.** Privacy-preserving LBS system model

During services, the client submits a query request to the LBS provider for seeking the suitable points of interest. To support the real LBS scenario, we adopt a polynomial evaluation algorithm<sup>1</sup> to select the best points for the client.

<sup>1</sup> To simply our description, we present the protocol by the evaluation algorithm in Eq. 2. It is worth to note that it is easy to extend our scheme to the polynomial evaluation algorithm over the multiple inputs for the real-world LBS applications.



$$Eva(P, Q) = \alpha \times dis(P, c) + \beta \times rating_P, \alpha < 0 \text{ and } \beta > 0. \quad (2)$$

Suppose a client  $c$  issues a query  $Q(c, \alpha, \beta)$ , where  $\alpha, \beta$  are the coefficients of the inputs distance and rating score,  $dis(P, c)$  is the distance between  $c$  and the point  $P$  in the road network,  $rating_P$  is  $P$ 's rating score. The LBS provider collects the rating scores from the clients and generates the rating score  $rating_P$  for every point  $P$ . In our scheme, the parameters  $\alpha$  and  $\beta$  are decided by the querier. If the querier more concerns with the distance, he will select a bigger  $\alpha$  value. Otherwise, he will select a bigger  $\beta$  value.

To protect the enterprise's interests, the LBS provider encrypts the point location and the rating score before uploading. Moreover, the LBS encrypts the query request from the client to generate a query trapdoor to protect the client's privacy. By the proposed encryption scheme, the CSP executes the polynomial evaluation over the encrypted data. While the LBS provider receives the encrypted results from the CSP, it decrypts and returns the results to the client.

## 4.2 Threat Model

For a privacy-preserving LBS application, it should protect the *data privacy* and the *client privacy*. The privacy violation could come from three aspects, i.e., the dataset of POIs, the encrypted point data and the query trapdoors. In general, the LBS provider will encrypt the POI description information using a block cipher such as AES [14]. Therefore, it is safe to claim that the privacy of the dataset of POIs itself is well protected. So, we focus on the privacy of other two aspects, i.e., the *point privacy* and the *trapdoor privacy*.

**Point Privacy:** The LBS provider encrypts the location and rating score to protect point privacy. The point privacy is twofold. Firstly, the cloud server should not learn the rating score through analyzing the encrypted point data. Secondly, since the point location is public and fixed, the location privacy in our scheme means that the CSP could not know the exact point location.

**Trapdoor Privacy:** The trapdoor is generated by the LBS provider to allow the CSP to execute polynomials over the encrypted point data. Intuitively, the trapdoor contains the query information but in an encrypted form. The trapdoor privacy means that the cloud server should learn nothing about the client from it, including his location and the polynomial evaluation function he decided.

## 5 Privacy-Preserving Polynomial Evaluation Algorithm

### 5.1 Overview

For  $n, N_L, N_r \in \mathbb{N}$ , we write  $[n]$  to denote the set of integers  $\{1, \dots, n\}$ . Let  $[N_L], [N_r]$  be the message spaces of point's location and rating score, and  $F : \{0, 1\}^\lambda \times \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda$  be a secure pseudorandom function (PRF) [8]. We design an intersection-based method to describe the point location in the road network. By this method, the LBS provider uses a number  $dis \in [N_L]$  to describe

the distance between a point to the specific intersection. The LBS provider and the CSP store the road network in the city as the basic information. Given a road network  $RN(I, R)$ ,  $I = \{I_1, I_2, \dots\}$  and  $R = \{r_1, r_2, \dots\}$  represent the set of intersections and roads respectively. The road  $r(r_{id}, I_j)$  is identified by the road id  $r_{id}$  and described by one of its intersection  $I_j$ . Given a point  $P(r_i, dis)$  which is on the road  $r(r_i, I_j)$ ,  $I_j$  is  $r_i$ 's intersection and  $dis$  describes the distance between  $P$  and  $I_j$ . Figure 2 illustrates the example of road network. Based on the road network, the LBS provider stores the distances between any two intersections. For a querier  $c$ , we map him to the nearest point  $c'(r_i, dis)$  on the road  $r(r_i, I_j)$  in which  $dis$  represents the distance between  $c'$  and  $r_i$ 's intersection  $I_j$ . We will detail the data encryption and storage mechanism in Sect. 5.3.

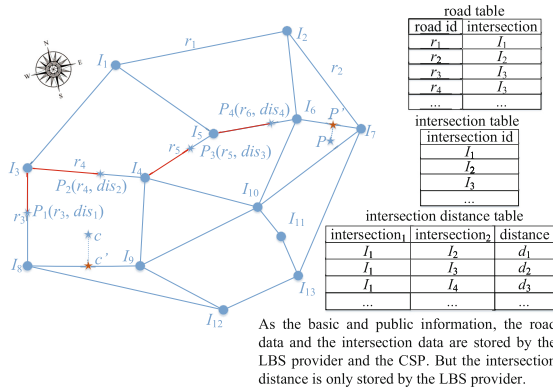


Fig. 2. Data storage over the road network

### 5.2 Homomorphic and Order-Revealing Encryption

In this paper, we propose a novel homomorphic and order-revealing encryption algorithm to design the privacy-preserving polynomial evaluation algorithm over road network. It consists of four polynomial-time algorithms as below:

- $\text{Setup}(1^\lambda) \rightarrow \{F, \text{CMP}, N, sk\}$  is run by the LBS provider to initialize the system. It first constructs a secure pseudorandom function  $F : \{0, 1\}^\lambda \times \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda$  and an additive homomorphic algorithm CMP as Definition 1 for the system. The LBS provider stores the secret key of CMP locally. Then, it takes a security parameter  $\lambda$  as input and outputs the secret key  $sk$ . The setup algorithm samples a PRF key  $k \xleftarrow{R} \{0, 1\}^\lambda$  for  $F$ , and a uniform random permutation  $\pi : [N] \rightarrow [N]$ , in which  $[N] = \max\{N_L, N_r\}$  denotes the message space of the plaintext. The secret key  $sk = (k, \pi)$ .
- $\text{Encrypt}_L(sk, m) \rightarrow ct_L$  is the left encryption algorithm run by the LBS provider. Given the secret key  $sk = (k, \pi)$ , the  $\text{Encrypt}_L$  algorithm outputs the left ciphertext for the message  $m$  as:

$$ct_L = (F(k, \pi(m)), \pi(m)). \tag{3}$$

- $\text{Encrypt}_R(sk, m) \rightarrow ct_R$  is the right encryption algorithm run by the LBS provider. Given the secret key  $sk = (k, \pi)$ , the  $\text{Encrypt}_R$  algorithm first samples a random element  $r \xleftarrow{R} \{0, 1\}^\lambda$ . Then, for each  $i \in [N]$ , it computes the value as

$$v_i = \text{CMP}(\pi^{-1}(i), m) + H(F(k, i), r) \bmod p, \tag{4}$$

in which  $H : \{0, 1\}^\lambda \times \{0, 1\}^\lambda \rightarrow \mathbb{Z}_p$  is a one way hash function. Finally, it outputs the right ciphertext for  $m$  as  $ct_R = (r, v_1, v_2, \dots, v_N)$ .

- $\text{Compute}(ct_L^{m_1}, ct_R^{m_2}) \rightarrow result$  is run by the CSP. It takes  $m_1$ 's left ciphertext  $ct_L^{(m_1)}$  and  $m_2$ 's right ciphertext  $ct_R^{(m_2)}$  as inputs and outputs  $result = \text{CMP}(m_1, m_2)$ , in which  $m_1, m_2$  are encrypted by the same key  $sk = (k, \pi)$ . The  $\text{Compute}$  algorithm first parses  $ct_L^{m_1} = (k', h) = (F(k, \pi(m_1)), \pi(m_1))$  and  $ct_R^{m_2} = (r, v_1^{m_2}, v_2^{m_2}, \dots, v_N^{m_2})$ , then outputs the result as:

$$result = v_h - H(k', r) \bmod p. \tag{5}$$

*Proof (correctness).* Let  $sk = (k, \pi) \leftarrow \text{Setup}(1^\lambda)$ , and any  $m_1, m_2 \in [N]$  be encrypted with  $sk$ ,  $ct_L^{m_1} = (k', h) = (F(k, \pi(m_1)), \pi(m_1)) \leftarrow \text{Encrypt}_L(sk, m_1)$  and  $ct_R^{m_2} = (r, v_1^{m_2}, \dots, v_N^{m_2}) \leftarrow \text{Encrypt}_R(sk, m_2)$ . Then, we have

$$\begin{aligned} result &= v_h - H(k', r) \\ &= \text{CMP}(\pi^{-1}(h), m_2) + H(F(k, \pi(m_1)), r) - H(F(k, \pi(m_1)), r) \\ &= \text{CMP}(\pi^{-1}(\pi(m_1)), m_2) + H(F(k, \pi(m_1)), r) - H(F(k, \pi(m_1)), r) \\ &= \text{CMP}(m_1, m_2) \in \mathbb{Z}. \end{aligned}$$

Note that, as defined in Definition 1,  $\text{CMP}(m_1, m_2) = E(m_1) \cdot E(m_2) = E(m_1 + m_2)$  provides the ciphertext of  $m_1 + m_2$ , so correctness follows.

### 5.3 Data Encryption and Storage

Based on the proposed homomorphic and order-revealing encryption scheme, we design the privacy-preserving polynomial evaluation algorithm over the location data and rating scores of the points in the road network. To protect data privacy, the LBS provider encrypts the point's data, i.e., the location and the rating score.

Let  $RN(I, R), I = \{I_1, \dots, I_{N_{inter}}\}, R = \{r_1, \dots, r_{N_{road}}\}$  be the road network in a city. In this work, an intersection  $I_{i(i \in [N_{inter}]})$  is described by its identifier  $I_i$ , and a road  $r_{i(i \in [N_{road}]})$  is described by the pair  $(r_i, I_j)$  in which  $r_i$  is the road identifier and  $I_j$  is one intersection of this road. We name  $I_j$  the road  $r_i$ 's intersection. We use  $dis(I_i, I_j)$  to represent the distance between  $I_i$  and  $I_j$ .

The LBS provider first calls  $\text{Setup}(1^\lambda) \rightarrow \{F, \text{CMP}, N, sk\}$  to initialize the system. Then, it generates the secret key  $sk_i = (k_i, \pi_i)$ , in which  $k_i$  is a secret PRF key of  $F$  and  $\pi_i$  is a uniform random permutation  $\pi_i : [N] \rightarrow [N]$ , for every intersection  $I_i$ .

In this work, we assume that all the points are on the roads. If a point is not on a road, we map it to the nearest point on the road as shown in Fig. 2. We use the road and the distance between the point and the road's intersection to describe

a point. Given a point  $P(r_i, dis_P, rating_P)$ ,  $dis_P \in [N_L]$ ,  $rating_P \in [N_r]$ , it means that  $P$  is on the road  $r_i$ , the distance between  $P$  and  $r_i$ 's intersection is  $dis_P$ , and the rating score of  $P$  is  $rating_P$ . We design an encryption algorithm  $\text{Encrypt}(P) \rightarrow ct_P$  by which the LBS provider encrypts  $P$  to protect its privacy.

Let  $P(r_i, dis_P, rating_P)$  be on the road  $r_i(r_i, I_j)$  in which  $I_j$ 's secret key is  $sk_j = (k_j, \pi_j)$ , the LBS provider selects a random element  $r \xleftarrow{R} \{0, 1\}^\lambda$  and encrypts the point data by the  $\text{Encrypt}_R$  algorithm with the key  $sk_j$  as Eq. 6. For a new point, the system sets its rating score  $rating_P$  as a default value.

$$\begin{aligned} ct_R(dis_P) &= \text{Encrypt}_R(sk_j, dis_P) = (r, v_1^d, v_2^d, \dots, v_{N_L}^d), \\ v_i^d &= \text{CMP}(\pi_j^{-1}(i), dis_P) + H(F(k_j, i), r) \bmod p \quad i \in [N_L], \\ ct_R(rating_P) &= \text{Encrypt}_R(sk_j, rating_P) = (r, v_1^R, v_2^R, \dots, v_{N_r}^R), \\ v_i^R &= \text{CMP}(\pi_j^{-1}(i), rating_P) + H(F(k_j, i), r) \bmod p \quad i \in [N_r]. \end{aligned} \quad (6)$$

By  $\text{Encrypt}(P) \rightarrow ct_P$ , the LBS provider outputs the ciphertext for the point  $P(r_i, dis_P, rating_P)$  as  $ct_P = (r_i, ct_R(dis_P), ct_R(rating_P))$ . In this work, we assume that the point's location will not be changed. During service, the LBS will periodically update the point's rating score<sup>2</sup>.

#### 5.4 Generation of Query Trapdoor

While a client  $c$  attempts to retrieve the suitable points around him,  $c$  generates a query request  $\mathcal{Q}(loc_c, \alpha, \beta)$  and asks the LBS provider to execute the polynomial evaluation  $f = \alpha \times dis(P, c) + \beta \times rating_P$  for him. In our scheme, we design a probabilistic trapdoor generation algorithm  $\text{TrapdoorGen}(sk, \mathcal{Q}) \rightarrow \mathcal{T}_{\mathcal{Q}}$  to protect the client's privacy. The LBS provider calls  $\text{TrapdoorGen}(sk_j, \mathcal{Q}) \rightarrow \mathcal{T}_{\mathcal{Q}}$  to generate the query trapdoor to evaluate all the points  $P(r_i, dis)$  where  $r_i$ 's secret key is  $sk_j(k_j, \pi_j)$ . After receiving the query request  $\mathcal{Q}(loc_c, \alpha, \beta)$ , the LBS provider generates the query trapdoor  $\mathcal{T}_{\mathcal{Q}}$  as follows:

1. If  $c$  is not on a road, the LBS provider maps him to the nearest node on the road  $r_i$ . The LBS provider transforms the query request  $\mathcal{Q}$  to  $\mathcal{Q}(r_i, dis_c, \alpha, \beta)$ , in which  $dis_c$  is the distance between  $c$  and  $r_i$ 's intersection  $I_i$ .
2. Given the distance between  $I_i$  and  $I_j$   $dis(I_i, I_j)$ , the LBS provider calculates the distance between  $c$  and  $I_j$  as  $dis(c, I_j) = dis(I_i, I_j) + dis_c$ . Then, the LBS provider encrypts  $dis(c, I_j)$  by  $\text{Encrypt}_L$  algorithm with  $I_j$ 's secret key  $sk_j(k_j, \pi_j)$  as

$$ct_L(dis_{\mathcal{Q}}) = (F(k_j, \pi_j(dis(c, I_j))), \pi_j(dis(c, I_j))). \quad (7)$$

3. The LBS provider selects a random number  $r_{\mathcal{Q}} \in [N_r]$  and encrypts it by  $\text{Encrypt}_L$  algorithm with  $sk_j$  as

$$ct_L(r_{\mathcal{Q}}) = (F(k_j, \pi_j(r_{\mathcal{Q}})), \pi_j(r_{\mathcal{Q}})). \quad (8)$$

<sup>2</sup> In this paper, we don't discuss how does the LBS provider decide the rating score from the users' responses. There are many mature technologies to address this issue.

4. The LBS provider selects a random number  $r' \in \mathbb{Z}_p$  and outputs the query trapdoor  $\mathcal{T}_Q = (I_j, ct_L(dis_Q), ct_L(r_Q), r'\alpha, r'\beta)$ .

### 5.5 Query of Privacy-Preserving Polynomial Evaluation

After receiving the trapdoor  $\mathcal{T}_Q = (I_j, ct_L(dis_Q), ct_L(r_Q), r'\alpha, r'\beta)$  from the LBS provider, the CSP calls  $\text{Query}(\mathcal{T}_Q, P) \rightarrow \omega_P$  to execute query over  $P$ 's ciphertext  $ct_P(r_i, ct_R(dis_P), ct_R(rating_P))$  which intersection also is  $I_j$  as follows:

1. Let the set  $S_Q$  be  $S_Q = \{P | ct_P = (r_i, ct_R(dis_P), ct_R(rating_P)), r_i = (r_i, I_j)\}$ . The CSP executes the query  $\mathcal{T}_Q$  over every point  $P \in S_Q$ .
2. By the Compute algorithm, the CSP first parses

$$\begin{aligned} (k'_{dis}, h_{dis}) &= ct_L(dis_Q) = (F(k_j, \pi_j(dis(c, I_j))), \pi_j(dis(c, I_j))) \\ (k'_{rating}, h_{rating}) &= ct_L(r_Q) = (F(k_j, \pi_j(r_Q)), \pi_j(r_Q)). \end{aligned}$$

3. Given  $ct_R(dis_P) = (r, v_1^d, \dots, v_{N_L}^d)$  and  $ct_R(rating_P) = (r, v_1^r, \dots, v_{N_r}^r)$ , the CSP computes

$$\begin{aligned} \mu_P &= v_{h_{dis}}^d - H(k'_{dis}, r) \bmod p, & \nu_P &= v_{h_{rating}}^R - H(k'_{rating}, r) \bmod p \\ \omega_P &= \mu_P^{r'\alpha} \cdot \nu_P^{r'\beta} \end{aligned}$$

4. Remove  $P$  from  $S_Q$ . If  $S_Q \neq \emptyset$ , pick another point  $P' \in S_Q$  and go back to the step 3, otherwise the CSP returns the result  $(\omega_P)_{P \in S_Q}$  to the LBS provider.
5. The LBS provider decrypts  $(\omega_P)_{P \in S_Q}$  as  $Eva_P = D(\omega_P)$  by CMP algorithm and sorts them. Finally, the LBS provider returns the top ones to the client.

*Proof (correctness).* Let  $sk_j = (k_j, \pi_j) \leftarrow \text{Setup}(1^\lambda)$  be the secret key of the intersection  $I_j$ ,  $ct_L(dis_Q) \leftarrow \text{Encrypt}_L(sk_j, dis(c, I_j))$ ,  $ct_P \leftarrow \text{Encrypt}_R(sk_j, dis_P)$ , and  $ct_R(dis_P) = (r, v_1^d, \dots, v_{N_L}^d)$ . Then, we have

$$\begin{aligned} \mu_P &= v_{h_{dis}}^d - H(k'_{dis}, r) \bmod p \\ &= \text{COM}(\pi_j^{-1}(h_{dis}), dis_P) + H(F(k_j, h_{dis}), r) - H(F(k_j, \pi_j(dis(c, I_j))), r) \\ &= \text{COM}(\pi_j^{-1}(\pi_j(dis(c, I_j))), dis_P) + H(F(k_j, \pi_j(dis(c, I_j))), r) \\ &\quad - H(F(k_j, \pi_j(dis(c, I_j))), r) \\ &= \text{COM}(dis(c, I_j), dis_P) \end{aligned}$$

Similarly, we also have that  $\nu_P = \text{COM}(r_Q, rating_P)$ .

Based on the homomorphic property of COM algorithm, we have

$$\begin{aligned} \omega_P &= \mu_P^{r'\alpha} \cdot \nu_P^{r'\beta} = \text{COM}(dis(c, I_j), dis_P)^{r'\alpha} \cdot \text{COM}(r_Q, rating_P)^{r'\beta} \\ &= \text{COM}(r'\alpha \times dis(c, I_j), r'\alpha \times dis_P) \cdot \text{COM}(r'\beta \times r_Q, r'\beta \times rating_P) \\ &= E(r'\alpha \times dis(c, I_j) + r'\alpha \times dis_P + r'\beta \times r_Q + r'\beta \times rating_P). \end{aligned}$$

Then, the LBS provider can decrypt the query results  $(\omega_P)_{P \in S_Q}$  and sort them as below:

$$\begin{aligned} D(\omega_P) &= D(\mu_P^{r'\alpha} \cdot \nu_P^{r'\beta}) = D(E(r'\alpha dis(c, I_j) + r'\alpha dis_P + r'\beta r_Q + r'\beta rating_P)) \\ &= r'\alpha(dis(c, I_j) + dis_P) + r'\beta(r_Q + rating_P), \end{aligned}$$

in which  $dis(c, I_j) + dis_P$  describes the distance between the querier  $c$  and  $P$ . Obviously, if  $D(\omega_{P_1}) > D(\omega_{P_2})$ , we have that  $P_1$ 's evaluation value is larger than  $P_2$ 's evaluation value by the polynomial evaluation function  $Eva(P, c) = \alpha \times dis(P, c) + \beta \times rating_P$ . So, the correctness follows.

*Remark 1.* In the correctness proof, we declare that  $dis(c, I_j) + dis_P$  equals to the distance between  $c$  and  $P$ . Note that it exists some errors in this measure. For the example in Fig. 2, the shortest path from  $c'$  to  $P_1$  is  $c' \rightarrow I_8 \rightarrow P_1$ . Since  $P_1$ 's intersection is  $I_3$ ,  $dis(c, I_j) + dis_P$  describes the path length of  $c' \rightarrow I_8 \rightarrow I_3 \rightarrow P_1$ . The measure error is  $O(length)$  in which  $length$  represents the length of road. So, we can divide a physical road into several short ones to reduce this measure error. Moreover, we can store a physical road by two visual roads with its two intersections respectively. For a query, we map the querier into two visual roads. By this method, we can remove this measure error completely. But, it also will double the costs of storage and computation. Therefore, in a real LBS application, the LBS provider could extend our scheme to find a balance between the costs and the accuracy.

## 5.6 Influence Model

By the basic scheme introduced above, the CSP need execute the query on all the intersections. This query mode will consume a lot of computation resources. We introduce an influence model to extend our scheme.

**Point Influence Model:** Executing the query on the distant intersections is useless, so we evaluate the influence range of the point and store the point on the intersections in its influence range.

Given the evaluation function  $Eva(P, c) = \alpha \times dis(P, c) + \beta \times rating_P$ , the parameters  $\alpha \in [range_{min}^\alpha, range_{max}^\alpha]$ ,  $\beta \in [range_{min}^\beta, range_{max}^\beta]$ , the maximal acceptable distance  $dis_{max}$ , and the minimal acceptable evaluation value  $s_{min}$ , we define the point  $P$ 's influence range as the set of  $S_{range}^P = \{I_i | dis(I_i, P) < dis_{max}, range_{max}^\alpha \times dis(I_i, P) + range_{max}^\beta \times rating_P > s_{min}\}$ . By the point influence model, the LBS provider encrypts and stores  $P$  on all the intersections in its influence range  $S_{range}^P$ . While a client  $c$  issues a query  $Q(r_i, dis_c, \alpha, \beta)$ , the LBS provider only needs to generate the query trapdoor on  $I_i$  which is  $r_i$ 's intersection. Because,  $I_i$  has stored all the candidates for the querier whose intersection also is  $I_i$ , CSP only needs to execute the query on  $I_i$ .

**Querier Influence Model:** Except for the point influence model, we introduce a querier influence model to limit the query range to improve the query efficiency. Given the same parameter setting with those in the point influence model, we

define the querier  $c$ 's influence range as the set of  $S_{range}^Q = \{I_i | dis(I_i, c) < dis_{max}\}$ . By the querier influence model, the LBS provider generates the query trapdoor  $\mathcal{T}_Q(r_i, dis_c, \alpha, \beta)$  on all the intersections in  $c$ 's influence range rather than executing the query on all the intersections in the road network.

By the influence model, the LBS provider is able to find a tradeoff in the storage, communication and query efficiency. With the point influence model, the CSP need store the multiple copies for a point on the intersections in its influence range, and the CSP only executes one-time query on the querier's intersection. With the querier influence model, the CSP does not store the redundant copies for a point, but it also need execute the queries on all the intersections in the querier's influence range.

## 6 Performance Analysis

### 6.1 Communication Cost

For the query request  $\mathcal{Q}(loc_c, \alpha, \beta)$ , the LBS provider generates the query trapdoor  $\mathcal{T}_Q = (I_j, ct_L(dis_Q), ct_L(r_Q), r'\alpha, r'\beta)$ . The size of  $\mathcal{Q}(loc_c, \alpha, \beta)$  is  $|S_l| + 2|S_{int}|$ , where  $S_l$  is the size of an element describing the user location, and  $|S_{int}|$  is the size of an integer. The size of  $\mathcal{T}_Q$  is  $|S_{id}| + 2(|S_F| + |S_{int}|)$ , where  $S_{id}$  is the size of the intersection identify,  $S_F$  is the size of one element in  $\{0, 1\}^\lambda$ .

In the basic scheme, the LBS provider need generate the query trapdoor on all the intersections. So, the communication costs for a query is  $Count_{int}|S_{id}| + 2Count_{int}(|S_F| + |S_{int}|)$ , in which  $Count_{int}$  is the number of intersections in the road network. In the improved scheme of the point influence model, the LBS provider only generates a query trapdoor on the intersection of the querier. In this case, the communication costs for a query is  $|S_{id}| + 2(|S_F| + |S_{int}|)$ . In the improved scheme of the querier influence model, the communication costs for a query is  $Count_{range}^{int}|S_{id}| + 2Count_{range}^{int}(|S_F| + |S_{int}|)$ , where  $Count_{range}^{int}$  represents the number of intersections in the querier's influence range.

### 6.2 Computation Cost

The computation cost of generating the query trapdoor by  $\text{TrapdoorGen}(sk_j, \mathcal{Q}) \rightarrow \mathcal{T}_Q$  algorithm is  $T_{add} + 2T_F + 4T_\pi + 2T_{mul}$  time, where  $T_{add}$ ,  $T_F$ ,  $T_\pi$  and  $T_{mul}$  represent one addition operation, one pseudorandom operation, one uniform random permutation operation and one multiplication operation.

During the query, for each point in the set  $S_Q$ , it takes  $T_{add} + T_{hash}$  for both distance and rating score according to Eq. 5 and  $2T_{exp} + T_{mul}$  for generating  $\omega_P$ . After receiving the results, the LBS provider takes  $T_D$  for each record and  $T_{sort}$  to return the top results, where  $T_D$  represents one decryption operation and  $T_{sort}$  represents one quick-sort operation. Therefore, the total computation costs for a query is  $|S_Q|(2T_{add} + 2T_{hash} + 2T_{exp} + T_{mul} + T_D) + T_{sort}$ . In the improved scheme, we reduce the computation costs by reducing the scale of  $S_Q$ .

## 7 Security Analysis

In this section, we prove that our scheme is secure under the random oracle model. Let  $\Pi = (\text{Setup}, \text{Encrypt}_L, \text{Encrypt}_R, \text{Compute})$  be the encryption scheme defined in Sect. 5.3. Let  $\mathcal{A} = (\mathcal{A}_1, \dots, \mathcal{A}_q)$  be an adversary for some  $q = \text{poly}(\lambda)$ , and let  $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \dots, \mathcal{S}_q)$  be a simulator. We adopt the similar experiments  $\text{REAL}_{\mathcal{A}}(\lambda)$  and  $\text{SIM}_{\mathcal{A}, \mathcal{S}}(\lambda)$  defined in [6]. We say that  $\Pi$  is a secure encryption scheme if for all polynomial-size adversaries  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_q)$ , there exists a polynomial-size simulator  $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_q)$  such that the two distributions  $\text{REAL}_{\mathcal{A}}(\lambda)$  and  $\text{SIM}_{\mathcal{A}, \mathcal{S}}(\lambda)$  are computationally indistinguishable.

Suppose the adversary has not issued an encryption query for a message  $m \in [N]$ . We prove that the adversary's view in the experiment is independent of  $f(\pi(m))$  in which  $f$  is a random function. Consider the ciphertext  $ct' = (ct'_L, ct'_R)$  the adversary obtains when it requests an encryption of some messages  $m' \neq m$ . Since  $\pi$  is a permutation,  $\pi(m') \neq \pi(m)$ . Next, because  $f$  is a random function,  $f(\pi(m'))$  is independent of  $f(\pi(m))$ . So, we conclude that the components of  $ct'_L$  are distributed independently of  $f(\pi(m))$ .

Consider  $ct'_R = (r', v'_1, v'_2, \dots, v'_N)$  for  $m'$ . Since  $r'$  is sampled uniformly at random from  $\{0, 1\}^\lambda$ , it is distributed independently of  $f(\pi(m))$ . And  $\forall i \in [N]$ ,  $v'_i = \text{CMP}(\pi^{-1}(i), m') + H(f(i), r')$ . The value of  $\text{CMP}(\pi^{-1}(i), m')$  is independent of the function  $f$ . Similarly, the output of the random oracle on  $(f(i), r')$  is independent of its input, i.e., independent of  $f(\pi(m))$ . Therefore, the components of  $ct'_R$  are distributed independently of  $f(\pi(m))$ .

Finally, the responses from the random oracle are distributed independently of  $f$ . Now, we let  $z_1, z_2, \dots, z_l, l = \text{poly}(\lambda)$  be the adversary's queries on the random oracle before it requests for an encryption of  $m$ . So, each  $z_i$  must be chosen independently of  $f(\pi(m))$ . Since  $f$  is a random function, the probability that there exists  $z_i = (f(\pi(m)), y)$  for any  $y$  is at most  $l/2^\lambda = \text{negl}(\lambda)$ . Because  $F$  in our scheme is a secure random function, we conclude that the proposed encryption scheme is secure under the random oracle model.

## 8 Experimental Study

### 8.1 Experiment Setup

We utilize jPaillier to implement the proposed homomorphic and order-revealing encryption in this paper. All the experiments are conducted under Ubuntu with an Intel 2.5 GHz processor and 16 GB memory. We use the representative real-world data set for California road network with California's points of interest<sup>3</sup>. The size of POI data set is 104,770 and the size of road network's intersections is 21,048. We compare our scheme with the PIR-based LBS query protocols in [12, 17, 18]. As most of recent works didn't provide personalized services, we fix the rating score parameter in query and simulate the polynomial evaluation scheme only considering the distance inputs. In data preparation stage, we randomly

<sup>3</sup> <https://www.cs.utah.edu/~lifeifei/SpatialDataset.htm>.



selects a rating score for each POI data, which is normally distributed from 1 to 5. In experiments, we randomly choose 100 points in the road network to issue the query request, in which  $\alpha$  is a random value while  $\beta$  is a fixed value.

## 8.2 Time Cost of Encrypting Point Data

In our scheme, the LBS provider encrypts the point data before outsourcing. We measure the time for preprocessing data with respect to the scale of data, which ranges from 10,000 to 100,000. We compare the time cost of encrypting data in our work with those in Yi et al. [17], Yi et al. [18] and Paulet et al. [12]. As the experimental results shown in Fig. 3, the encryption efficiency of our scheme is comparable with others. By the basic scheme, the time costs of encrypting point data are from 1.816 to 10.455 s, which depends on the scale of POI dataset. When we choose the point influence model, it costs the extra time for encrypting POI data on the intersections in the influence range. It is consistent with the theoretical analysis.

## 8.3 Time Cost of Generating Query Trapdoor

In our scheme, the computation cost of generating the query trapdoor includes the cost of building a query in the road network and the cost of generating the trapdoor. We carry out the experiment to evaluate the computation costs at the LBS provider for generating the query trapdoor. We compare the time cost of our scheme with those in [12, 17, 18]. The experimental results in Fig. 4 show that our scheme can effectively reduce the time cost at the LBS provider because only pseudorandom function and uniform random permutation are executed. Moreover, the influence model can further reduce the time costs by only generating the query trapdoors on the intersections in the influence range. While the same operations in [12, 17, 18] are executed on the whole map.

## 8.4 Time Cost of Executing Polynomial Evaluation

We carry out the experiments to measure the time costs for executing a polynomial evaluation. The experimental results are shown in Fig. 5. By the basic scheme, the query request needs to be executed on all the intersections in the road network. The CSP finishes the query from 4.856 to 12.950 s. After using the influence model, our scheme can provide the more efficient polynomial evaluation. The time costs are from 1.132 to 2.650 s with the point influence model and from 2.105 to 3.523 s with the querier influence model. Our scheme achieves a better query performance than others. Moreover, our scheme can support the personalized polynomial evaluation, which still can not be implemented by the existing researches.

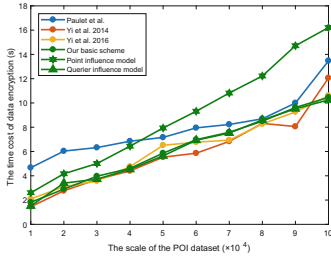


Fig. 3. Time cost for data encryption

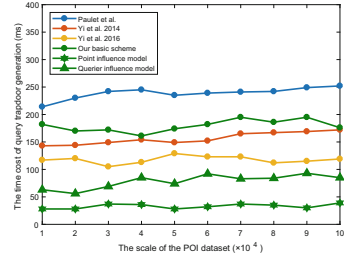


Fig. 4. Time cost for trapdoor generation

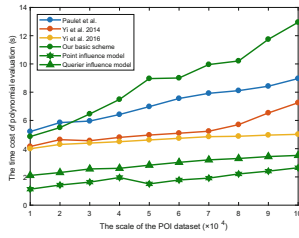


Fig. 5. Time cost of query

## 9 Conclusion

In this paper, we examined the practical problem of executing the privacy-preserving polynomial evaluation function over the outsourced data in the road network. We further identified three practical requirements for this problem. To fulfill these requirements, we proposed a novel privacy-preserving polynomial evaluation algorithm based on order-revealing encryption. For practical purposes, we extended our scheme by introducing an influence model to reduce the overheads of computation and communication. We have proved that our scheme is secure for the LBS scenarios. We also carried out several experiments to show that the query processes are efficient and our scheme is appropriate for using in the cloud-based LBS systems.

**Acknowledgements.** This work is supported in part by the National Natural Science Foundation of China under grants 61572378, the Natural Science Foundation of Hubei Province under grant 2017CFB420.

## References

1. Boneh, D., Lewi, K., Raykova, M., Sahai, A., Zhandry, M., Zimmerman, J.: Semantically secure order-revealing encryption: multi-input functional encryption without obfuscation. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 563–594. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-46803-6\\_19](https://doi.org/10.1007/978-3-662-46803-6_19)
2. Elmehdwi, Y., Samanthula, B.K., Jiang, W.: Secure k-nearest neighbor query over encrypted data in outsourced environments. In: Proceedings of ICDE, pp. 664–675 (2014)
3. Ghinita, G., Kalnis, P., Khoshgozaran, A., Shahabi, C., Tan, K.L.: Private queries in location based services: anonymizers are not necessary. In: Proceedings of SIGMOD, pp. 121–132 (2008)
4. Ghinita, G., Kalnis, P., Skiadopoulos, S.: PRIVE: anonymous location-based queries in distributed mobile systems. In: Proceedings of WWW, pp. 371–380 (2007)
5. Hu, H., Xu, J., Ren, C., Choi, B.: Processing private queries over untrusted data cloud through privacy homomorphism. In: Proceedings of ICDE, pp. 601–612 (2011)
6. Kevin, L., David J., W.: Order-revealing encryption: new constructions, applications, and lower bounds. In: Proceedings of CCS, pp. 1167–1178 (2016)
7. Liu, S., et al.: Efficient query processing with mutual privacy protection for location-based services. In: Proceedings of DASFAA, pp. 299–313 (2016)
8. Oded, G., Shafi, G., Silvio, M.: How to construct random functions. *J. ACM* **33**(4), 792–807 (1986)
9. Palanisamy, B., Liu, L.: MobiMix: protecting location privacy with mix-zones over road networks. In: Proceedings of ICDE, pp. 494–505 (2011)
10. Papadopoulos, S., Bakiras, S., Papadias, D.: Nearest neighbor search with strong location privacy. *VLDB Endow* **3**(1–2), 619–629 (2010)
11. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). [https://doi.org/10.1007/3-540-48910-X\\_16](https://doi.org/10.1007/3-540-48910-X_16)
12. Paulet, R., Kaosar, M.G., Yi, X., Bertino, E.: Privacy-preserving and content-protecting location based queries. *IEEE Trans. Knowl. Data Eng.* **26**(5), 1200–1210 (2014)
13. Song, W., Wang, B., Wang, Q., Shi, C., Lou, W., Peng, Z.: Publicly verifiable computation of polynomials over outsourced data with multiple sources. *IEEE Trans. Inf. Forensics Secur.* **12**(10), 2334–2347 (2017)
14. Vincent, R., Joan, D.: Advanced encryption standard. In: Federal Information Processing Standards Publications, pp. 19–22 (2001)
15. Wong, W.K., Cheung, D.W.I., Kao, B., Mamoulis, N.: Secure kNN computation on encrypted databases. In: Proceedings of SIGMOD, pp. 139–152 (2009)
16. Yao, B., Li, F., Xiao, X.: Secure nearest neighbor revisited. In: Proceedings of ICDE, pp. 733–744 (2013)
17. Yi, X., Paulet, R., Bertino, E., Varadharajan, V.: Practical k nearest neighbor queries with location privacy. In: Proceedings of ICDE, pp. 640–651 (2014)
18. Yi, X., Paulet, R., Bertino, E., Varadharajan, V.: Practical approximate k nearest neighbor queries with location and query privacy. *IEEE Trans. Knowl. Data Eng.* **28**(6), 1546–1559 (2016)

19. Yiu, M.L., Assent, I., Jensen, C.S., Kalnis, P.: Outsourced similarity search on metric data assets. *IEEE Trans. Knowl. Data Eng.* **24**(2), 338–352 (2012)
20. Zhou, C., Wang, T., Jiang, W., Tian, H.: Practical k nearest neighbor query scheme with two-party guarantees in road networks. In: *Proceedings of TrustCom*, pp. 1316–1321
21. Zhou, L., Zhu, Y., Castiglione, A.: Efficient k-NN query over encrypted data in cloud with limited key-disclosure and offline data owner. *Comput. Secur.* **69**, 84–96 (2017)
22. Zhu, Y., Huang, Z., Takagi, T.: Secure and controllable k-NN query over encrypted cloud data with key confidentiality. *J. Parallel Distrib. Comput.* **89**, 1–12 (2016)

# **Recommendation**



# AdaCML: Adaptive Collaborative Metric Learning for Recommendation

Tingting Zhang<sup>1</sup>, Pengpeng Zhao<sup>1</sup>(✉), Yanchi Liu<sup>2</sup>, Jiajie Xu<sup>1</sup>, Junhua Fang<sup>1</sup>,  
Lei Zhao<sup>1</sup>, Victor S. Sheng<sup>3</sup>, and Zhiming Cui<sup>4</sup>

<sup>1</sup> Institute of Artificial Intelligence, School of Computer Science and Technology,  
Soochow University, Suzhou, China

ttzhang7061@stu.suda.edu.cn,

{ppzhao, xujj, jhfang, zhao1}@suda.edu.cn

<sup>2</sup> Rutgers University, New Brunswick, USA

yanchi.liu@rutgers.edu

<sup>3</sup> University of Central Arkansas, Conway, USA

ssheng@uca.edu

<sup>4</sup> Suzhou University of Science and Technology, Suzhou, China

zmcui@mail.usts.edu.cn

**Abstract.** User preferences are dynamic and diverse in real world, while historical preference of a user may not be equally important as current preference when predicting future interests. As a result, learning the evolving user representation effectively becomes a critical problem in personalized recommendation. However, existing recommendation solutions often use a fixed user representation, which is not capable of modeling the complex interests of users. To this end, we propose a novel metric learning approach named Adaptive Collaborative Metric Learning (AdaCML) for recommendation. AdaCML employs a memory component and an attention mechanism to learn an *adaptive user representation*, which dynamically adapts to locally activated items. In this way, implicit relationships of user-item pairs can be better determined in the metric space and users' interests can be modeled more accurately. Comprehensive experimental results demonstrate the effectiveness of AdaCML on two datasets, and show that AdaCML outperforms competitive baselines in terms of Precision, Recall, and Normalized Discounted Cumulative Gain (NDCG).

**Keywords:** Recommender systems · Attention mechanism · Metric learning

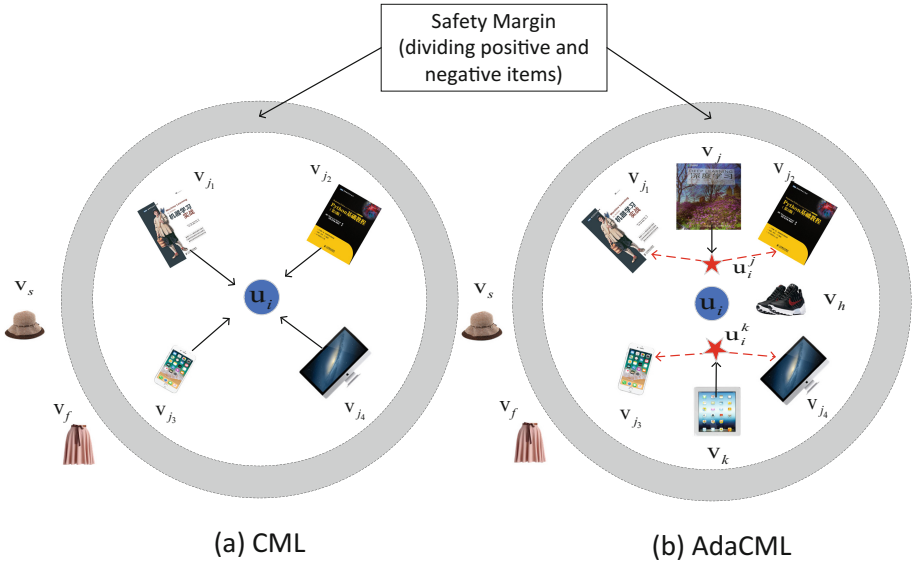
## 1 Introduction

With the explosive growth of e-commerce and social media, we are living in an era of information explosion. Recommender Systems (RS) can alleviate the dilemma of information overload by delivering more relevant products to us. Collaborative Filtering (CF) is one of the most successful ways to build recommender systems and has received significant success in the past decades [1, 11, 17].

© Springer Nature Switzerland AG 2019

G. Li et al. (Eds.): DASFAA 2019, LNCS 11447, pp. 301–316, 2019.

[https://doi.org/10.1007/978-3-030-18579-4\\_18](https://doi.org/10.1007/978-3-030-18579-4_18)



**Fig. 1.** The left image shows the standard collaborative metric learning method, and the right image denotes the adaptive user representation that can dynamically adapt to related items.

Specifically, it considers users’ historical interactions and makes recommendations based on their similar preferences. Among all models of CF, Matrix Factorization (MF) [11] remains the most popular one, which tries to characterize both users and items in a low-dimensional latent vector space inferred from a historical user-item interaction matrix. However, MF captures users’ preferences for items by performing the dot product operation that violates the triangular inequality, resulting in suboptimal performance [9]. The triangle inequality states that for any three objects, the sum of any two pairwise distances should be greater than or equal to the remaining pairwise distance [9]. This implies that if  $x$  is close to  $y$  and  $z$ , then  $y$  is also close to  $z$ . Therefore, the MF method based on the dot product operation can only capture the local preferences of the observed data (e.g.,  $x$  is close to  $y$  and  $z$ ), but fails to capture the potential preference information (e.g.,  $y$  is also close to  $z$ ). Hsieh et al. [9] proposed Collaborative Metric Learning (CML) scheme that uses Euclidean distance to measure user-item relationships, which follows triangle inequality. CML not only captures user-item relationships, but also learns user-user similarity and item-item similarity in vector space. This collaborative metric learning (CML) method [9] has demonstrated highly competitive performance on many benchmark datasets.

Although CML obtains promising performance, it still faces some challenges. CML often projects users and items as fixed low-dimensional representation vectors into a user-item joint space. However, a fixed user representation greatly limits the ability of the CML method to model the diversities of users’ interests.

This leads to a lack of flexibility in the CML model, which is unable to approach different candidate items discriminatively. In fact, users prefer different kinds of items, and only a subset of users' historical items is relevant to the candidate products. For instance, a user has previously purchased books, shoes, and badminton rackets, and then the user may buy badminton. His/Her purchase of badminton is more related to the badminton rackets than the books or the shoes previously purchased. Hence, it is difficult to use a fixed user representation to model the diversities of users' interests. As shown in Fig. 1, a user  $u_i$  liked a list of different kinds of items  $\{v_{j_1}, v_{j_2}, v_{j_3}, v_{j_4}\}$ . The user  $u_i$  and these items are represented as a latent vector in a user-item metric space. Since similar items are projected to similar locations, these items that user  $u_i$  liked form two clusters according to the types of items. Figure 1(a) shows that the essential idea of CML is to pull positive items (such as, items  $\{v_{j_1}, v_{j_2}, v_{j_3}, v_{j_4}\}$ ) closer to the user  $u_i$  and push the negative items (i.e., items that the user did not like, such as, item  $v_s$  and  $v_f$ ) away until they are beyond the safety margin. Given a list of candidate items  $\{v_j, v_k, v_h\}$ , the CML method with a fixed user representation will recommend the nearest unvisited items, such as item  $v_h$ , as shown in Fig. 1(b). Although item  $v_h$  is closer to user  $u_i$  than item  $v_j$  and item  $v_k$ , it is obvious that item  $v_j$  and item  $v_k$  are more reliable and valuable recommendations. Moreover, CML pursues performance while ignoring interpretability and significant insight, which is also a common problem of other methods [8, 14].

To address the above problems, we propose an Adaptive Collaborative Metric Learning (AdaCML) method for recommendation. AdaCML uses a memory module and a novel attention mechanism to learn an *adaptive user representation* in a metric space. In this work, we assume that each user prefers different kinds of items and only a subset of the user's purchased products can reveal whether the user is interested in a candidate product. Hence, for different candidate items, an *adaptive user representation* adaptively selects locally activated items from user's historical items. As shown in Fig. 1(b), for the candidate item  $v_j$ , an *adaptive user representation*  $\mathbf{u}_i^j$  is obtained by dynamically weighting the vectors of items  $v_{j_1}$  and  $v_{j_2}$ . Similarly, another *adaptive user representation*  $\mathbf{u}_i^k$  is more related to items  $v_{j_3}$  and  $v_{j_4}$ . In this way, our model not only improves the performance but also enables interpretable recommendation results.

The main contributions of this paper are summarized as follows:

- The proposed AdaCML novelly applies an *adaptive user representation* to metric learning. The *adaptive user representation* can be dynamically adapted to locally activated items.
- We use a memory component and an attention mechanism to flexibly acquire *adaptive user representations* based on candidate items, which not only greatly improve the accuracy of the model, but also provide insight and interpretability for the model.
- We evaluate our AdaCML model on two real-world datasets. Our experimental results show that AdaCML outperforms various benchmarks with a significant margin.



The rest of the paper is organized as follows. We first define the problem of this paper in Sect. 2 and then present the framework of AdaCML in Sect. 3. After reporting our experimental study in Sect. 4, we introduce related work in Sect. 5. We finally conclude this paper in Sect. 6.

## 2 Problem Definition

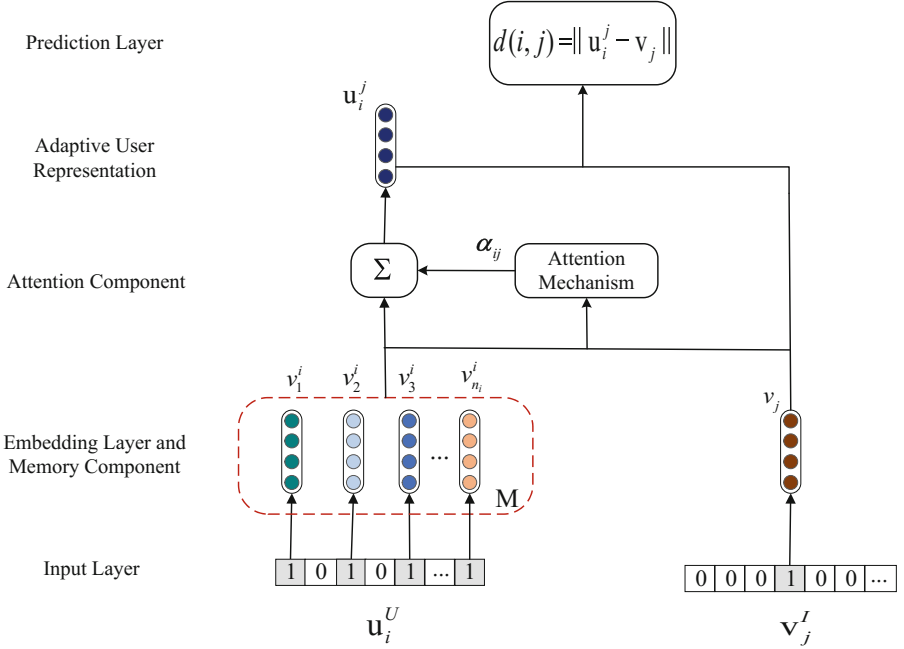
In this section, we first introduce notations used in this paper, and then define the recommendation problem of our study. We apply bold capital letters (e.g.,  $\mathbf{X}$ ) and bold lowercase letters (e.g.,  $\mathbf{x}$ ) to represent matrices and vectors, respectively. Also, we employ non-bold letters (e.g.,  $x$ ,  $X$ ) to denote scalars, and squiggle letters (e.g.,  $\mathcal{X}$ ) to denote sets. In this paper, we denote a set of users as  $\mathcal{U} = \{u_1, u_2, \dots, u_N\}$  and an item set as  $\mathcal{I} = \{v_1, v_2, \dots, v_M\}$ , where  $N$  and  $M$  denote the number of users and items, respectively. For a user  $u_i \in \mathcal{U}$ , let  $\mathcal{R}_i^+ = \{v_1, v_2, \dots, v_{n_i}\}$  denotes the set of  $n_i$  items liked by user  $u_i$  and  $\mathcal{R}_i^-$  denotes the remaining items. Then, given user set  $\mathcal{U}$  and item set  $\mathcal{I}$ , for each user  $u_i \in \mathcal{U}$ , the recommendation task is defined as recommending a list of items from  $\mathcal{R}_i^-$  that the user  $u_i$  may be interested in.

## 3 Adaptive Collaborative Metric Learning (AdaCML)

In this section, we first propose the overall architecture of AdaCML, and then introduce the details of each layer. Lastly, we will introduce the optimization details and time complexity analysis of AdaCML.

### 3.1 The Model Architecture

Figure 2 illustrates the architecture of the Adaptive Collaborative Metric Learning (AdaCML). The AdaCML model is based on a personalized memory module and a novel adaptive attention mechanism. Intuitively, users' interests are diverse, and users' historical records contain different types of items. The core issue of AdaCML is to design an *adaptive user representation* to characterize appetites of a user more accurately when the user interacts with different candidate items. Our model consists of five components, input layer, embedding layer, attention component, *adaptive user representation* and prediction layer. Specifically, the bottom input layer consists of two sparse vectors that denote user and item, respectively, where each element of each vector is either 0 or 1. Note that unlike the CML model that uses one-hot vector of each user as the input feature, AdaCML leverages multi-hot historical item IDs of each user as input features. Then the sparse vector is fed to the embedding layer, and each element with a value of 1 in the sparse vector corresponds to a dense vector. That is, feeding a user sparse vector to the embedding layer returns multiple dense vectors. Hence, we apply a memory matrix  $\mathbf{M}$  to store the representation of each item in each user's historical records, so the memory matrix  $\mathbf{M}$  naturally reflects the user's preferences. To design an *adaptive user representation*,



**Fig. 2.** The architecture of the adaptive collaborative metric learning.

we introduce an item-level attention mechanism on the memory matrix  $\mathbf{M}$  to capture items in  $\mathcal{R}_i^+$  relevant to item  $v_j$ . Here, we regard each item as a unit and model the impact of previously purchased items on the target item. After getting the *adaptive user representation*, we use the metric learning method to train and optimize our model AdaCML. Next, we will introduce each part of our model in details.

**Input Layer.** As shown in Fig. 2, the bottom input layer consists of two sparse vectors  $\mathbf{u}_i^U$  and  $\mathbf{v}_j^I$  that describe user  $u_i$  and item  $v_j$ , respectively. Moreover, the elements of the input vector of user  $u_i$  and item  $v_j$  are composed of 0 or 1. For  $\mathbf{u}_i^U$ , item IDs visited by user  $u_i$  are represented as 1, and the unvisited items are 0; for  $\mathbf{v}_j^I$ , where the index corresponding to item  $v_j$  is represented as 1 and the rest of positions are 0.

**Embedding Layer and Memory Component.** Above the input layer is the embedding layer that projects sparse representation to dense representation. The obtained each dense representation can be regarded as the latent feature vector for an item. Hence, for an item  $v_j$ , we formulate the item embedding vector as

$$\mathbf{v}_j = \mathbf{Q}^T \mathbf{v}_j^I, \quad (1)$$

where  $\mathbf{Q} \in \mathbb{R}^{K \times M}$ , denoting the latent vector matrix for all items.  $M$  is the number of items and  $K$  is the dimensionality of the user/item latent vector.

Since the user sparse vector  $\mathbf{u}_i^U$  contains all item IDs previously accessed by the user  $u_i$ , this sparse vector input embedding layer will get multiple dense vectors. Therefore, we apply a memory module to store these dense feature vectors. This memory matrix  $\mathbf{M}$  is constructed as

$$\mathbf{M} = \{\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_{n_i}\}, \quad (2)$$

where  $\mathbf{M} \in \mathbb{R}^{K \times n_i}$ . Each column of matrix  $\mathbf{M}$  corresponds to a representation of an item in  $\mathcal{R}_i^+$ , so the matrix  $\mathbf{M}$  reflects all interests of a specific user.

**Attention Component.** As mentioned in the introduction section, a fixed user vector representation may fail to convey diverse appetites of a user, resulting in the limitation of the representation ability of a model. Although the memory matrix  $\mathbf{M}$  stores the information of all historical items purchased by each user, the goal of the attention mechanism is to selectively pick items that are closely related to the target item from  $\mathbf{M}$  and then aggregate the representation of informative items to characterize the interests of a specific user [3]. Hence, given the user-item pair,  $(u_i, v_j)$ , the AdaCML model uses an inner product to learn the similarity between the  $q$ -th item of the memory component and the target item  $v_j$ . Each element of the similarity score  $\mathbf{w}_{ij}$  is defined as follows:

$$w_{ij}^q = \mathbf{m}_q^T \mathbf{v}_j, \forall q = 1, 2, \dots, n_i, \quad (3)$$

where  $\mathbf{m}_q \in \mathbb{R}^{K \times 1}$  is the  $q$ -th column vector of memory matrix  $\mathbf{M}$ . In order to normalize the similarity score  $\mathbf{w}_{ij}$  to the attention weights distribution, we adopt the softmax function, which is a common practice in the attention network. We use  $\alpha_{ij}^q$  to denote the  $q$ -th element of the attention vector  $\boldsymbol{\alpha}$  as:

$$\alpha_{ij}^q = \frac{\exp(w_{ij}^q)}{\sum_{h \in \mathcal{R}_i^+} \exp(w_{ij}^h)}. \quad (4)$$

Note that the larger  $\alpha_{ij}^q$  indicates the higher relevance between a target item  $v_j$  and a corresponding item  $v_q$  in  $\mathbf{M}$ .

**Adaptive User Representation.** In order to generate a dynamic and *adaptive user representation*, we use the attention vector  $\boldsymbol{\alpha}$  to calculate a weighted sum representation of  $\mathbf{M}$ , i.e., adaptively selecting valuable pieces from the memory matrix  $\mathbf{M}$ . Hence, the *adaptive user representation*  $\mathbf{u}_i^j$  is derived by weighting  $\mathbf{M}$  according to the candidate item  $v_j$  as:

$$\mathbf{u}_i^j = \sum_{q \in \mathcal{R}_i^+} \mathbf{m}_q \alpha_{ij}^q, \quad (5)$$

where  $\mathbf{m}_q$  corresponds to the  $q$ -th column of the memory matrix  $\mathbf{M}$ , and  $\mathcal{R}_i^+$  denotes historical items liked by user  $u_i$ . It's worth noting that the proposed user representation is non-fixed and dependent on the candidate item  $v_j$ .

**Prediction Layer.** After computing an *adaptive user representation*  $\mathbf{u}_i^j$  and an item vector  $\mathbf{v}_j$ , we learn these vectors according to their Euclidean distance.

Hence, when making predictions, we feed the final *adaptive user representation*  $\mathbf{u}_i^j$  and the target item embedding  $\mathbf{v}_j$  into the Euclidean distance:

$$d(i, j) = \|\mathbf{u}_i^j - \mathbf{v}_j\|, \quad (6)$$

where the *adaptive user representation*  $\mathbf{u}_i^j$  is dependent on the attributes of the target item  $v_j$  and the user  $u_i$ 's historical preferences. Here, the Euclidean function is used to measure the distance of user  $u_i$  and item  $v_j$ . It's more likely that the user  $u_i$  prefers item  $v_j$  if the distance between them is closer.

Overall, our proposed model AdaCML has the following advantages. First, considering the case where a given user's interests are diverse, we can leverage all items rated by the user to gain additional information. Second, with an attention mechanism, we allow each member of historical records of a user to contribute in *adaptive user representation* decision, where the contribution of each member is dependent on its feature and the property of a target item. Finally, an *adaptive user representation* can dynamically adapt to the user's interests based on the characteristics of the target item in an interpretable manner.

### 3.2 Model Inference

Finally, our model AdaCML adopts the hinge loss for optimization. For each positive user-item pair,  $(u_i, v_j)$ , we randomly select several items from the user's unvisited records to form several negative user-item pairs. Similar to positive examples, negative pairs are represented as corresponding users and items hidden vectors in the same way. The hinge loss is defined as follow:

$$L = \sum_{(i,j) \in \mathcal{S}} \sum_{(i,j') \notin \mathcal{S}} \sigma_{ij} [\lambda + d(i, j)^2 - d(i, j')^2]_+, \quad (7)$$

where  $\mathcal{S}$  is a set of positive user-item pairs,  $j$  is an item user  $u_i$  preferred,  $j'$  is an item he/she did not like,  $[z]_+ = \max(z, 0)$  denotes the standard hinge loss, and  $\lambda > 0$  is the safety margin size that separates positive pairs and negative pairs. It is worth noting that because the user's representation depends on the attributes of the target item, positive user-item pairs do not share the same representation with negative user-item pairs. Namely, given a positive user-item pair  $(u_i, v_j)$  and a negative user-item pair  $(u_i, v_{j'})$ , the corresponding user feature vector are  $\mathbf{u}_i^j$  and  $\mathbf{u}_i^{j'}$ , respectively.

Besides, our model, like CML, also uses a rank-based weighting scheme called Weighted Approximate-Rank Pairwise (WARP) loss, which was proposed by Weston et al. [16], to penalize positive items at a lower rank. As a result, we penalize the positive item  $v_j$  based on its rank by setting

$$\sigma_{ij} = \log(\text{rank}_d(i, j) + 1), \quad (8)$$

where  $\text{rank}_d(i, j)$  denotes the rank of item  $v_j$  in the recommended list of user  $u_i$ . At last, we employ regularization to constrain all the item embedding within a unit ball, i.e.,  $\|\mathbf{v}\|^2 \leq 1$ , to prevent overfitting and ensure the robustness of the learned metric.

### 3.3 Time Complexity Analysis

In this subsection, we will analyze the time complexity of the AdaCML model. As we described in Sect. 3.1, the *adaptive user representation* component of AdaCML directly reflects the time cost of AdaCML in testing, i.e., Eq. (5). For a user-item pair,  $(u_i, v_j)$ , the main time cost of an *adaptive user representation*  $\mathbf{u}_i^j$  comes from the attention network. We use  $K$  to denote the embedding size,  $|\mathcal{R}_i^+|$  to denote the number of historical interactions of user  $u_i$ , and  $a$  to denote the attention size. In this work,  $a$  is equal to  $K$ . And then we can obtain that the time complexity of each element of the similarity score  $\mathbf{w}_{ij}$  is  $O(aK)$ . However, the similarity score  $\mathbf{w}_{ij}$  includes  $|\mathcal{R}_i^+|$  elements, so after using the softmax function to normalize  $\mathbf{w}_{ij}$ , the time complexity of the attention weight  $\alpha_{ij}$  becomes  $O(aK|\mathcal{R}_i^+|)$ . Finally, we use the weighted sum of the attention weight and the memory matrix to get the user adaptive representation. Hence, for each user, the time score of AdaCML is  $O(aK|\mathcal{R}_i^+|^2)$ . However, considering that the denominator term is shared across the computation of all items in  $|\mathcal{R}_i^+|$ , we only need to calculate it once and cache it for all items in  $|\mathcal{R}_i^+|$ . Therefore, the final time complexity of evaluating an AdaCML prediction is  $O(aK|\mathcal{R}_i^+|)$ .

## 4 Experiments

In this section, we will conduct experiments with the aim of answering the following research questions:

- RQ1.** How does our proposed model AdaCML perform, compared with state-of-the-art recommendation methods?
- RQ2.** How do the hyper-parameters influence the performance of AdaCML?
- RQ3.** Can the AdaCML model improve the interpretability of recommendation?

### 4.1 Experimental Settings

**Datasets.** We will study the performance of AdaCML on the Amazon dataset [7, 13]. Amazon<sup>1</sup> is an e-commerce platform and is widely used for product recommendation evaluation. We used two subsets of the Amazon review data as two datasets, namely Instant Video and Automotive. The Instant Video dataset contains users' ratings and review texts for the instant videos, while the Automotive dataset provides users' purchase records of various auto parts and accessories. Since we focus on the implicit feedback of users, we transformed the detailed ratings in the two datasets into 0 or 1, which indicates whether the user has rated the item. The statistics of the datasets are summarized in Table 1.

**Baselines.** We will compare our model AdaCML with following five state-of-the-art baseline models.

<sup>1</sup> <http://jmcauley.ucsd.edu/data/amazon/>.

**Table 1.** Datasets statistics

Dataset	Users	Items	Ratings	Sparsity
Instant video	1352	7785	16863	99.84%
Automotive	2928	1835	12962	99.76%

- **BPR** [14] is a strong baseline for building CF recommendation from implicit feedback data, which proposes a pair-wise optimization method to model the relative preference of users.
- **MLP** [8] applies a multi-layer perceptron above user and item embeddings to learn the scoring function. It is a standard neural network model with ReLU activates.
- **NeuMF** [8] is a unified framework fusing Matrix Factorization and Multi-Layer Perceptron. This model combines the linearity of Matrix Factorization and the non-linearity of Deep Neural Networks for modeling user-item interactions. Following the original work, we use the proposed tower structure.
- **MARS** [18] is a combination of Convolutional Neural Network (CNN) and attention mechanisms to learn an *adaptive user representation*. Since our experiment does not consider the text content of each item, we randomly initialize the item embedding based on Gaussian distribution to replace the operation of CNN extracting item features.
- **CML** [9] is a metric-based collaborative recommendation method, which solves the limitation of the inner product operate and obeys the triangle inequality.

**Evaluation Metrics and Implementation Details.** We divided each user’s ratings into trainsets/testsets in a 70%/30% split. To evaluate these models described above, we repeated each individual model five times, and reported the average result of each model. In addition, we evaluated AdaCML and all baselines in terms of three different metrics, i.e., Precision, Recall, and Normalized Discounted Cumulative Gain (NDCG). We optimized all models with the Adam Optimizer. The learning rate of all models are tuned amongst  $\{0.1, 0.01, 0.005, 0.001\}$  and the batch size is fixed to 512. Without a special mention in this paper, we fixed the embedding size of all models to 32 for a fair comparison. Also, we set the margin size  $\lambda = 1.5$  for AdaCML model, and  $\lambda = 2$  for CML model on two datasets. We will investigate the influences of embedding size and margin size in Sect. 4.3. For AdaCML, the number of negative samples is fixed at 10, while the number of negative samples of all the baselines follows the setting in corresponding original papers. Both AdaCML and CML apply regularization by normalizing all user/item embedding to be constrained within the Euclidean ball. Also, for NeuMF and MLP, we employ a three-layer MLP that follows the original paper settings with the shape of (32, 16, 8). The MARS model can be understood as a variant of BPR, which takes the *user adaptive representation* into account, but it still uses dot product operations to capture the relationships between users and items.

**Table 2.** Experimental results of AdaCML and baselines. The best performance of each column (the larger the better) is in bold, and the second best is underlined.

Dataset	Method	@5			@10			@20		
		Recall	Precision	NDCG	Recall	Precision	NDCG	Recall	Precision	NDCG
Instant Video	BPR	0.01518	0.01102	0.03582	0.02595	0.00954	0.04635	0.04493	0.00825	0.05974
	MLP	0.01835	0.01295	0.03407	0.03161	0.01128	0.04726	0.04688	0.00871	0.05752
	NeuMF	<u>0.02686</u>	<u>0.01975</u>	<u>0.05117</u>	0.04347	<u>0.01631</u>	0.06503	0.07160	0.01333	0.08204
	CML	0.02573	0.01795	0.05069	<u>0.04413</u>	0.01614	<u>0.06749</u>	<u>0.07720</u>	<u>0.01423</u>	<u>0.08804</u>
	MARS	0.01870	0.01302	0.04478	0.02693	0.01002	0.05397	0.04228	0.00803	0.06495
	AdaCML	<b>0.03587</b>	<b>0.02626</b>	<b>0.07298</b>	<b>0.06553</b>	<b>0.02374</b>	<b>0.09645</b>	<b>0.10633</b>	<b>0.01923</b>	<b>0.11800</b>
Automotive	BPR	0.02023	0.01031	0.03240	0.03342	0.00856	0.04230	0.05298	0.00675	0.01198
	MLP	0.02412	0.01220	0.03873	0.03719	0.00927	0.04866	0.06770	0.00853	0.06476
	NeuMF	0.03930	0.01961	0.06037	0.06041	0.01508	0.07358	0.09445	0.01182	0.09003
	CML	0.03231	0.01600	0.04691	0.05456	0.01352	0.06131	0.08218	0.01032	0.07446
	MARS	<u>0.04892</u>	<u>0.02377</u>	<u>0.07137</u>	<u>0.07771</u>	<u>0.01913</u>	<u>0.09181</u>	<u>0.11810</u>	<u>0.01470</u>	<u>0.10942</u>
	AdaCML	<b>0.05584</b>	<b>0.02777</b>	<b>0.08087</b>	<b>0.09058</b>	<b>0.02249</b>	<b>0.10203</b>	<b>0.13654</b>	<b>0.01711</b>	<b>0.12161</b>

## 4.2 Performance Comparison (RQ1)

We compared the performance of AdaCML with five state-of-the-art baselines regarding Recall, Precision, and NDCG with cutoffs at 5, 10, and 20 on two datasets. Their overall recommendation performances are shown in Table 2. The experimental results reveal many interesting points as follows.

- Firstly, regardless of the datasets and the evaluation metrics, AdaCML always achieves the best performance, outperforming the state-of-the-art methods by a significant margin. This shows that by merging the *adaptive user representation* to metric learning, AdaCML can better model users' dynamic and diverse interests in a metric space, resulting in better recommendations.
- Secondly, pertaining to the performance of the baselines, we found that the performance of BPR is inferior on the two datasets. CML performs better than BPR. This indicates that modeling user-item relationships based on metric learning methods is more efficient than inner product. We can also see that NeuMF and CML perform similarly and outperform all other baselines on Instant Video. On the Automotive dataset, NeuMF also shows excellent performance and defeats most baselines except MARS. This shows that the NeuMF combines MF and MLP for a better performance. Moreover, we observed that MLP usually performs better than BPR, but falls short of CML on the two datasets in terms of Recall and Precision metrics. It is possible that NeuMF is a combination of multiple models to achieve well performance, and MLP alone may not yield good results. Besides, MARS performs better than BPR on the two datasets. This confirms that it is also effective to consider *adaptive user representation* in BPR.
- Finally, it is interesting to see that the performance of MARS is quite unstable. MARS performs quite well on the dataset Automotive while it does not perform well on Instant Video. This may be because the Instant Video is more relevant to real-time information, and only a small amount of items

previously accessed by the user are related to the current recommendation. However, AdaCML still achieves excellent results in this situation. This confirms the advantages of the metric learning method.

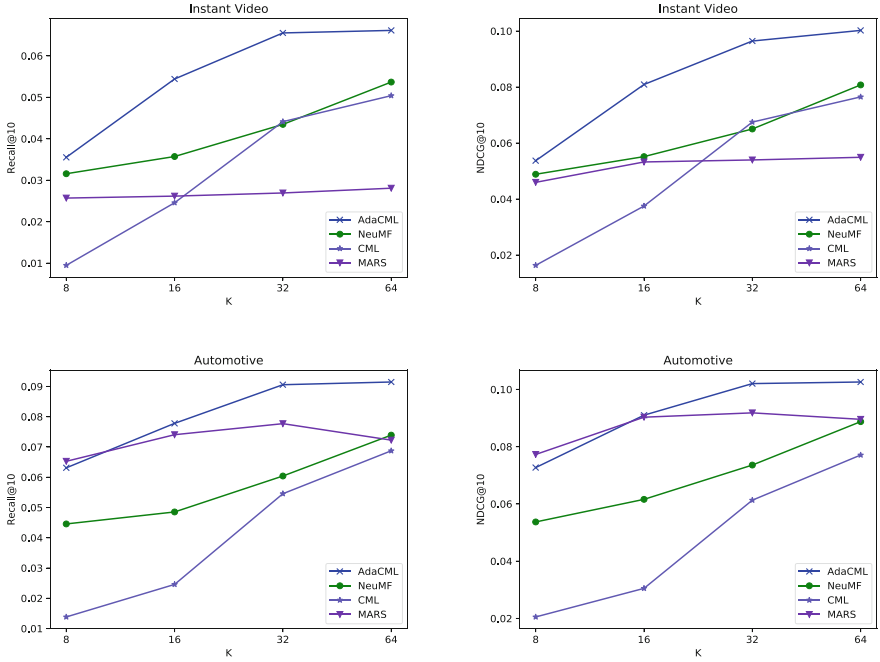


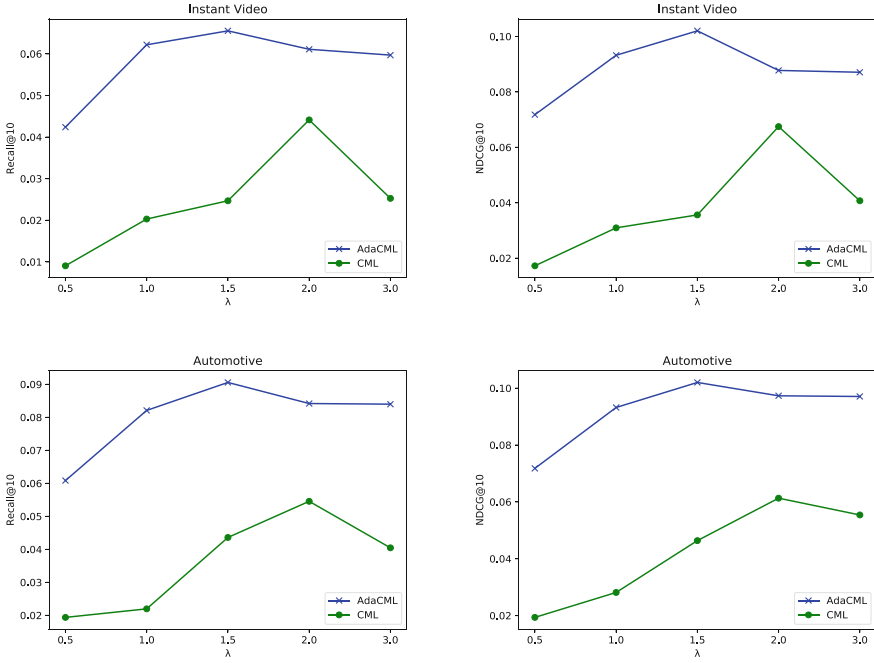
Fig. 3. Performance of these models under difference choices of the embedding size  $K$ .

### 4.3 Hyper-parameter Study (RQ2)

As we mentioned in Sect. 3, AdaCML has two crucial hyper-parameters (i.e., the embedding size  $K$  and the margin size  $\lambda$ ). In this subsection, we will investigate the impact of the two hyper-parameters respectively. We only show the experimental results of the Recall@10 and NDCG@10 because of the space limitation.

Figure 3 shows the performance of these methods at embedding size 8, 16, 32, and 64. Due to the poor performance of BPR and MLP, they are omitted in Fig. 3 to better highlight the performance differences among the rest of methods. On both datasets, AdaCML achieves the best performance under most cases, with the only exception of the embedding size setting as 8 on the dataset Automotive. This may be because the smaller embedding size is not enough to capture the complex relationships between users and items in the metric space. We also notice that CML performs extremely horribly with the embedding size setting as 8, which validates that a small hidden factor fails to represent users-items relationships. On the dataset Automotive, MARS reaches the peak when the embedding size is set as 32, followed by a degradation potentially due to





**Fig. 4.** Performance of AdaCML and CML under different choices of the margin size  $\lambda$ .

overfitting. Compared with other models, the fluctuation of the experimental results of MARS is slight. This confirms that the method based on the dot product operation is not enough to capture the implicit relationships between users and items. Last but not least, as the embedding size increases, the performance of most models on the two datasets increases. This is because larger dimensions could capture more hidden factors of users and items, and then achieve better performance.

Figure 4 shows the performance of AdaCML and CML at margin size 0.5, 1.0, 1.5, 2.0, and 3.0. We can see that regardless of the setting of the margin size  $\lambda$ , AdaCML always outperforms CML. As we can see from Fig. 4, when  $\lambda = 1.5$ , AdaCML achieves the best performance, while CML reaches the peak at  $\lambda = 2$ . We attribute the phenomenon to the trade-off between positive items and negative items: too small  $\lambda$  can hardly distinguish between the positive and negative products, while too large  $\lambda$  brings much more noises (negative products) than useful products (positive products) within the safety margin.

#### 4.4 The Interpretability of AdaCML (RQ3)

To gain a better insight into the interpretation ability of AdaCML, we conducted a qualitative experiment in this subsection. Table 3 shows the item-level attention weights concerning the items that users liked on the dataset Automotive. The

weights have been normalized to make a clear comparison. For example, for user #541, the target item #1486 is a positive example in the testset, while items #856, #85, and #237 are historical ones. We can see that AdaCML assigns higher weights on items #856 and #85, a lower weight on item #237, successfully evaluating the target item #1486 as the item desired by the user. However, for user #2048, the historical items #337 and #1014 have a higher weight, indicating that items #337 and #1014 are more relevant to the target item #397. To demonstrate the rationality, we further investigated the content of these items (i.e., the Automotive review text). Regarding user #541, we found that both the target item #1486 and the two higher attended items #856 and #85 are for better engine performances, while the lowest weight item #237 is about the windshield wiper. For user #2048, items #393 and #1014 with higher weights and the target item #397 are led lights, while the lowest weight item is Hoerr Racing Products. This is expected, because when predicting a user's preference on a target item, his/her historical items of the same category should have a larger impact than other less relevant items. Overall, this case study shows that AdaCML has great interpretability to explain its recommendation results.

## 5 Related Work

Our proposed AdaCML can be seen as an improvement of the popular Collaborative Metric Learning method (CML) [9]. It overcomes the drawbacks of CML by using a memory module and a novel attention mechanism to learn an *adaptive user representation* in a metric space. In this section, we will briefly discuss closely related work from two aspects, which are collaborative filtering and attention mechanism.

**Table 3.** A case study on the interpretability of AdaCML. The first column shows the user ID, the second column displays the target item ID, and the third column has three sub-columns, each of which contains a historical item and an attention weight, where the attention value is shown in parentheses.

User ID	Target item ID	Historical items		
		1	2	3
#541	#1486	#856 (0.398)	#85 (0.399)	#237 (0.203)
#2048	#397	#337 (0.393)	#1267 (0.239)	#1014 (0.368)

### 5.1 Collaborative Filtering

Collaborative Filtering (CF) is one of the most successful technique to build recommender systems, operating on explicit and implicit feedback data. Implicit

collaborative filtering is a method of learning user-item interactions from implicit data (e.g., buy or not buy, like or not), and also known as the one-class recommendation [7]. In implicit feedback data, user-item relationships are binary in nature (i.e., 1 if observed, and 0 otherwise). Bayesian Personalized Ranking (BPR) [14] is a well-known framework for disposing of the implicitness in CF. Instead of point-wise learning as in SVD [6], BPR uses a pair-wise learning algorithm to model the relative preferences of users, which makes the items that each user preferred more forward than those items he/she did not like. CML [9] is a recently proposed algorithm for CF and has demonstrated highly competitive performance on several benchmarks. CML learns a joint metric space that not only encodes users' preference for items but also learns user-user similarity and item-item similarity. Unlike most methods, CML uses Euclidean distance to measure the preferred relationships between users and items. CML minimizes the distance between each positive user-item interaction in Euclidean space. Hence, in this space, the items that users liked will keep coming close to users, and the uninterested items will be pushed further away. But CML still faces some challenges. Firstly, the representation of users is fixed and is insufficient to capture the complex and diverse interests. Secondly, CML lacks interpretability. To address these challenges, we propose AdaCML, which uses a memory module and a novel attention mechanism to learn an *adaptive user representation* in a metric space. It not only greatly improves the recommendation performance, but also provides the greater insight and the interpretability of recommendation. The details of AdaCML was explained in Sect. 3.

## 5.2 Attention Mechanism

Attention mechanism are prevalent in many domains, such as image/video caption [4, 5], machine translation [12], and so on. The attention mechanism originates from the neural science studies by empirically demonstrating that human usually focus on specific parts of the target entity rather than using all available information [10]. The key idea of attention is to learn or select the attentive weights from a set of inputs. Higher (lower) weights indicate that the relevant components (members) are more informative (less informative) to generate the end output. Recently, the attention mechanism has been widely wielded to more accurately represent features in recommender systems [2, 3, 15]. For example, Chen et al. [2] proposed an interesting attention-driven factor model for recommendation. It adopts the attention model to measure the relevance of each part of each item. To get the representation for a multimedia object (e.g., image or video), Chen et al. [3] aggregated its components (e.g., regions or frames) with an attention network, where a similar attention mechanism is applied to aggregate interacted items to get a user representation for making recommendation.

Moreover, in most models, the attention mechanism is usually applied to calculate the importance of each feature in the item content to the user, which does not consider candidate items. This method generally captures the user's general preference and lacks flexibility, causing weak performance and poor explanation. Zheng et al. [18] utilized CNN and the attention mechanism to adaptively

capture local crucial historical items based on the contents of candidate items. However, MARS [18] still uses inner product to portray user-item interactions, which violates the triangular inequality. Hence, we present an AdaCML model that follows the triangle inequality. Our experimental results show that it has a better performance in capturing the diverse interests of users by combining the metric learning method with the attention mechanism.

## 6 Conclusion

In this paper, we propose an Adaptive Collaborative Metric Learning (AdaCML) model to perform a personalized recommendation for users. As we know, users' interests are diverse and involved in various domains. Most existing methods often focus on learning a fixed and static user representation. This is not enough to model the diverse interests of users. To tackle this problem, we used a memory component and an attention mechanism to learn an *adaptive user representation*, and utilized a metric learning algorithm to capture the relationships of user-item, user-user, and item-item. To the best of our knowledge, AdaCML is the first model to apply an *adaptive user representation* to a metric learning algorithm. We conducted extensive experiments on two real-world datasets and demonstrated that AdaCML could consistently outperform the state-of-the-art models by a significant margin.

**Acknowledgments.** This research was partially supported by the NSFC (61876117, 61876217, 61872258, 61728205, 61772356), the Suzhou Science and Technology Development Program (SYG201803) and the Blockshine Technology Corp.

## References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.* **17**(6), 734–749 (2005)
2. Chen, J., Zhuang, F., Hong, X., Ao, X., Xie, X., He, Q.: Attention-driven factor model for explainable personalized recommendation. In: *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 909–912. ACM (2018)
3. Chen, J., Zhang, H., He, X., Nie, L., Liu, W., Chua, T.S.: Attentive collaborative filtering: multimedia recommendation with item-and component-level attention. In: *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 335–344. ACM (2017)
4. Chen, L., et al.: SCA-CNN: Spatial and channel-wise attention in convolutional networks for image captioning. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6298–6306. IEEE (2017)
5. Cho, K., Courville, A., Bengio, Y.: Describing multimedia content using attention-based encoder-decoder networks. *IEEE Trans. Multimedia* **17**(11), 1875–1886 (2015)
6. Golub, G.H., Reinsch, C.: Singular value decomposition and least squares solutions. *Numer. Math.* **14**(5), 403–420 (1970)

7. He, R., McAuley, J.: Ups and downs: modeling the visual evolution of fashion trends with one-class collaborative filtering. In: Proceedings of the 25th International Conference on World Wide Web, WWW 2016, pp. 507–517 (2016)
8. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web, WWW 2017, pp. 173–182 (2017)
9. Hsieh, C.K., Yang, L., Cui, Y., Lin, T.Y., Belongie, S., Estrin, D.: Collaborative metric learning. In: Proceedings of the 26th International Conference on World Wide Web, WWW 2017, pp. 193–201 (2017)
10. Itti, L., Koch, C., Niebur, E.: A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* **20**(11), 1254–1259 (1998)
11. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. *Computer* **42**(8), 30–37 (2009)
12. Luong, M.T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation (2015). arXiv preprint: [arXiv:1508.04025](https://arxiv.org/abs/1508.04025)
13. McAuley, J., Targett, C., Shi, Q., Van Den Hengel, A.: Image-based recommendations on styles and substitutes. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 43–52. ACM (2015)
14. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, pp. 452–461. AUAI Press (2009)
15. Vinh, T.D.Q., Pham, T.A.N., Cong, G., Li, X.L.: Attention-based group recommendation (2018). arXiv preprint: [arXiv:1804.04327](https://arxiv.org/abs/1804.04327)
16. Weston, J., Bengio, S., Usunier, N.: Large scale image annotation: learning to rank with joint word-image embeddings. *Mach. Learn.* **81**(1), 21–35 (2010)
17. Zhang, Y., Wang, H., Lian, D., Tsang, I.W., Yin, H., Yang, G.: Discrete ranking-based matrix factorization with self-paced learning. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 2758–2767. ACM (2018)
18. Zheng, L., et al.: Mars: memory attention-aware recommender system (2018). arXiv preprint: [arXiv:1805.07037](https://arxiv.org/abs/1805.07037)



# Adaptive Attention-Aware Gated Recurrent Unit for Sequential Recommendation

Anjing Luo<sup>1</sup>, Pengpeng Zhao<sup>1,5</sup>(✉), Yanchi Liu<sup>2</sup>, Jiajie Xu<sup>1,5</sup>, Zhixu Li<sup>1</sup>,  
Lei Zhao<sup>1</sup>, Victor S. Sheng<sup>3</sup>, and Zhiming Cui<sup>4</sup>

<sup>1</sup> Institute of Artificial Intelligence, School of Computer Science and Technology,  
Soochow University, Suzhou, China

ajluo@stu.suda.edu.cn, {ppzhao,xujj,zhixuli,zhaol}@suda.edu.cn

<sup>2</sup> Rutgers University, New Brunswick, USA

yanchi.liu@rutgers.edu

<sup>3</sup> University of Central Arkansas, Conway, USA

ssheng@uca.edu

<sup>4</sup> Suzhou University of Science and Technology, Suzhou, China

zmcui@mail.usts.edu.cn

<sup>5</sup> Neusoft Corporation, Shenyang, China

**Abstract.** Due to the dynamic and evolutionary characteristics of user interests, sequential recommendation plays a significant role in recommender systems. A fundamental problem in the sequential recommendation is modeling dynamic user preference. Recurrent Neural Networks (RNNs) are widely adopted in the sequential recommendation, especially attention-based RNN becomes the state-of-the-art solution. However the existing fixed attention mechanism is insufficient to model the dynamic and evolutionary characteristics of user sequential preferences. In this work, we propose a novel solution, Adaptive Attention-Aware Gated Recurrent Unit (3AGRU), to learn adaptive user sequential representations for sequential recommendation. Specifically, we adopt an attention mechanism to adapt the representation of user sequential preference, and learn the interaction between steps and items from data. Moreover, in the first level of 3AGRU, we construct adaptive attention network to describe the relevance between input and the candidate item. In this way, a new input based on adaptive attention can reflect users' diverse interests. Then, the second level of 3AGRU applies adaptive attention network to hidden state level to learn a deep user representation which is able to express diverse interests of the user. Finally, we evaluate the proposed model using three real-world datasets from various application scenarios. Our experimental results show that our model significantly outperforms the state-of-the-art approaches on sequential recommendation.

**Keywords:** Adaptive attention mechanism · GRU · Recommender system

## 1 Introduction

With the explosion of Web information, recommender system plays a more and more significant role in online services, where capturing users' preferences is critical. Due to the intrinsically dynamic and evolving characteristics of user interests, sequential recommendation has attracted a lot of attention in recommender systems. A fundamental problem in the sequential recommendation is how to model dynamic and evolutionary user preferences to satisfy user needs better [19].

For modeling sequential patterns, Factorizing Personalized Markov Chain (FPMC) model was proposed to factorize user-specific transition matrix by the Markov Chain (MC) [19]. A significant drawback of MC-based solutions is that they adopt the static representation for user's interests. With the success of neural networks in many application domains, recurrent neural networks (RNNs) are widely adopted in the sequential recommendation, such as session-based recommendation [8], next-basket and next-item recommendations [12, 13].

Besides the essential dynamic and evolutionary characteristics, the user's interests are also diverse (not singular) in the same period, and they usually involve multiple fields. For example, we may find that a user who likes reading books about deep learning also likes purchasing household appliances. Although various extensions of RNN, like LSTM and GRU, can better capture the long-term dependency of user preference, they assume temporal dependence has a monotonic change with each step. In other words, the current item is more significant than the previous one to predict the next one, which is not always true. Attention network based RNN can solve the above problem, where the attention mechanism can automatically assign different influences to previous items, and achieve the state-of-the-art performance [2, 12, 24].

However, the recommendation process can be too dynamic for the attention-based solutions to capture. A previous item may play a different role and exhibit different influences in choosing next items of different types due to the specialty. Nevertheless, existing attention-based RNN solutions use a fixed strategy to aggregate the influences of previous step items. As such, they are insufficient to capture the dynamic process of users' diverse sequential decision makings, resulting in a suboptimal solution. Let us illustrate the above problem with an example. Suppose a user bought three items e.g., Python book, iPad, RecommenderSystem book in a time order. Subsequently, the user purchases an iWatch, and the sequential history is finalized as Python book, iPad, RecommenderSystem book, iWatch. If we take the first three items as the context and the last one as the target to recommend, existing fixed attention-based methods may suggest books like deep learning books due to the more influence of book items. However, the choice of the target item iWatch may depend on the first item (iPad). In this case, recommender systems should pay more attention to the iPad when computing the score of candidate recommendation item iWatch, because iPad may be more related to the next choice iWatch. This example shows the influence of previous items may be more related to candidate items. Therefore, it may not be optimal and realistic to recommend next items with a fixed attention mechanism.

In this paper, to express the dynamics and diversity of users' interests, we develop an Adaptive Attention-Aware Gated Recurrent Unit model (3AGRU) for sequential recommendation. First of all, we leverage the strength of the recurrent architecture of GRU to capture complex long-term dependencies and that of the attention network to discover the local sequential pattern. More importantly, motivated by the observation of user behaviors, we facilitate GRU with a novel adaptive item-level attention mechanism to devise a deep adaptive user sequential interest representation. Unlike fixed attention-based user representations, adaptive user representations dynamically adapt to locally activated items. The first level of 3AGRU is conducted on the input level to make the hidden state stronger. We input a new input generated by current item and adaptive attention which considering the information of candidate item to GRU. The second level of 3AGRU is executed on the hidden state level to utilize adaptive attention network to learn a deep adaptive user representation. Accordingly, the hidden state strengthened by the adaptive input is further intensified, which is more in line with the users' interests. The contributions of this work can be summarized as follows:

- We introduce a novel adaptive attention-aware recurrent neural network for adaptive user representation, which adaptively combines both user's long-term and short-term preferences to generate a high-level hybrid representation of users.
- The adaptive contextual attention networks on input level and hidden state level are further integrated to improve the performance of sequential recommendation.
- We compare our model 3AGRU with state-of-the-art methods and verify the superiority of 3AGRU through quantitative analysis on three large real-world datasets. The experimental results reveal that our method is capable of leveraging user historical records effectively.

In the following part of the paper, we first introduce the related work in Sect. 2, and define the problem in Sect. 3. Then, we illustrate our framework in Sect. 4. In Sect. 5, we verify the effectiveness of our method with experimental results. Finally, the conclusions and outlooks of this work are presented in Sect. 6.

## 2 Related Work

Our Adaptive Attention-Aware Gated Recurrent Unit (3AGRU) is proposed for sequential recommendation. Therefore, in this section, we discuss related work from two aspects, i.e., general recommendation and sequential recommendation.

### 2.1 General Recommendation

General recommendation recommends items through modeling the users' general tastes from their historical interactions. The key idea is collaborative filtering (CF), which can be further categorized into memory-based CF and model-based



CF [21]. The memory-based CF provides recommendations by finding k-nearest-neighbours of users or items based on similarity [15], while the model-based CF tries to factorize the user-item correlation matrix for recommendation [9, 11]. [17] introduces weights to user-item pairs, and optimizes the factorization with both least-square and hinge-loss criteria. [18] optimizes the latent factor model with a pairwise ranking loss in a Bayesian framework. [26] optimizes cross-entropy loss between the true pairwise preference ranking and predicted pairwise preference ranking for each user. General recommendation can capture users' general taste, but can hardly adapt its recommendations directly to users' recent interactions without modeling sequential behaviors.

## 2.2 Sequential Recommendation

Sequential recommendation views the interactions of a user as a sequence and aims to predict which item the user will interact with next. A typical solution to this setting is to compute an item-to-item relational matrix, whereby the most similar (the nearest) items to the last interacted one are recommended to users. For example, Markov Chain based methods estimate an item-to-item transition probability matrix and use it to predict the probability of the next item given the last interaction of a user [19, 20]. [20] presents a recommender based on Markov decision processes and shows that a predictive Markov Chain model is effective for next basket prediction. [19] combines a factorization method and Markov Chains, using the factorization method to model the user general taste and Markov Chains to mine user sequential patterns. Hierarchical representation model combines the last action information with the general user interest to model user representations for next basket recommendation [22]. For these Markov Models, it is difficult to model the long-range dependence. Prod2Vec, inspired by word embedding technique [16], learns distributed item representations from the interaction sequences and uses them to compute a cosine similarity matrix [5]. [6] assumes that items are embedded into a "transition space" where each user is modeled by a translation vector.

Recently, Recurrent Neural Network (RNN), a state-of-the-art deep learning method for sequence modeling, is shown to be effective in capturing sequential user behavioral patterns [3, 8, 27]. Different from previous methods, applying RNN to sequential recommender introduces the capability of modeling the whole historical interactions. [8] implements an improved version of the GRU network for session-based recommendation, which utilizes a session-parallel mini-batch training process and employs ranking-based loss functions for learning the model. DREAM [25] utilizes pooling to summarize the basket of embedded items and then feeds into vanilla RNN to solve next basket recommendation. [10] integrates the RNN-based networks with knowledge base enhanced Key-Value Memory Network (KV-MN) to capture sequential user preference and attribute-level user preference. [14] explains a K-plet Recurrent Neural Network for accommodating multiple sequences jointly to capture global structure and localized relationships at the same time.

However, our proposed 3AGRU model differs from existing attention-based RNN solutions. These solutions use a fixed attention mechanism to aggregate influence of previous items while 3AGRU facilitates relevance between previous items and candidate items to devise a deep adaptive user sequential interest representation. Thus, our model 3AGRU can express the dynamics and diversity of users' interests.

### 3 Problem Statement

In this section, we first introduce basic notations that will be used in this paper. Let  $\mathcal{U} = \{u_1, u_2, \dots, u_{|\mathcal{U}|}\}$  denote a set of users and  $\mathcal{I} = \{i_1, i_2, \dots, i_{|\mathcal{I}|}\}$  denote a set of items, where  $|\mathcal{U}|$  and  $|\mathcal{I}|$  are the total numbers of users and items, respectively. Each user  $u$  is associated with a sequence of some items from  $\mathcal{I}$ ,  $\mathcal{I}^u = \{i_1^u, \dots, i_t^u, \dots, i_n^u\}$ , where  $i_t^u \in \mathcal{I}$  and  $n$  is the number of items interacted with user  $u$ . The index  $t$  for  $i_t^u$  denotes the relative time index, not the absolute timestamp.

With the above notations, we define the sequential recommendation task as follows. In this work, we focus on the case of implicit action feedback. Given a user  $u$ 's history transaction sequence  $I^u$ , we aim to predict a list of items that the user would probably interested in the near future.

## 4 Adaptive Attention-Aware Gated Recurrent Unit

In this section, we will display the process of our proposed Adaptive Attention-Aware Gated Recurrent Unit model (3AGRU) for sequential recommendation in details. Firstly, we introduce the basic GRU model. Secondly, we present the overall architecture of 3AGRU and then introduce the details of each layer of it. Finally, we offer the optimization procedures.

### 4.1 Gated Recurrent Unit

As a variant of LSTM, GRU solves the problem of long-term dependence of RNN well and simplifies the structure of LSTM. It contains a reset gate  $r_t$  and an update gate  $z_t$ . Besides, it has the candidate state  $\tilde{h}_t$  which uses  $r_t$  to control the inflow of the last hidden state containing previous information. If the reset gate is approximately zero, the last hidden state will be discarded.  $r_t$  determines how much information was forgotten in the past.  $h_t$  is the hidden state which uses  $z_t$  to update the last hidden state  $h_{t-1}$  and the  $\tilde{h}_t$ . The update gate  $z_t$  controls the importance of the previous hidden state at the current moment. The formulas are as follows.

$$r_t = \sigma(x_t W_{xr} + h_{t-1} W_{hr} + b_r) \quad (1)$$

$$z_t = \sigma(x_t W_{xz} + h_{t-1} W_{hz} + b_z) \quad (2)$$

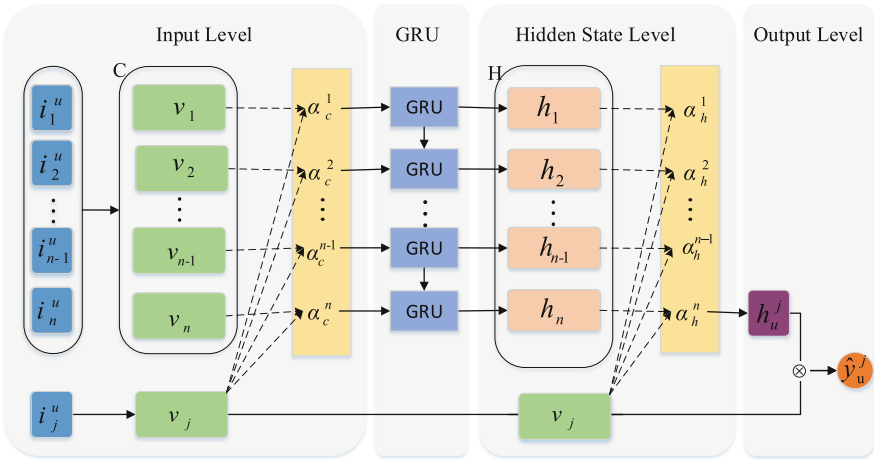
$$\tilde{h}_t = \tanh(x_t W_{xh} + r_t \odot h_{t-1} W_{hh} + b_h) \quad (3)$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t \tag{4}$$

In these formulas,  $x_t$  is the input vector while  $t$  is the time step,  $W_{xr}, W_{hr}, W_{xz}, W_{hz}, W_{xh}, W_{hh}, b_r, b_z, b_h$  represent the transition matrices and bias of the input and hidden levels in  $r_t$  and  $z_t$  respectively,  $\odot$  is the element-wise product between two vectors, and the sigmoid function  $\sigma(x)$  is used to do nonlinear projection. The last  $h_t$  is the final representation of the sequence.

### 4.2 3AGRU

The 3AGRU model is a hierarchical structure that consists of input level, GRU, hidden state level and output level. What we will do is to apply adaptive attention network to the input level and the hidden state level of GRU. The modeling architecture is shown in Fig. 1. In the input level, the sparse inputs are embedded into dense representations to get memory component  $C$  which contains information of all items of user  $u$  and  $v_j$  which is the representation of a candidate item  $i_j^u$ . Then the adaptive attention network is employed on  $C$  as the input of GRU. In this way, the input of GRU at each time step has different weights, and the larger the weight value is, the more similar  $i_j^u$  and the corresponding item are. In the hidden state level, the adaptive attention network is also applied to the set of the hidden state at each time step to get the new hidden state with different weights. The different weights denote the relevance between the candidate item and the hidden state of GRU at each time step. With the information passed to the final hidden state, the final hidden state is used to be the adaptive user representation. Finally, we use the final hidden state with attention to measure the user’s preference score over item  $i_j^u$ . Then, we will introduce each part of our model in detail.



**Fig. 1.** The architecture of 3AGRU. The weight in Adaptive Attention Network indicates the correlation between the corresponding input item and the candidate item.

**Adaptive Attention Network in Input Level.** The input level is composed of embedding layer and adaptive attention network. As we all known, both original input and candidate item have limited representation ability that is similar to discrete words symbols in natural language processing. Therefore, our model 3AGRU will use item embedding to project sparse inputs to dense representations. Formally, let  $C \in \mathbb{R}^{k \times n}$  represent dense input whose column corresponds to a representation of an item in  $\mathcal{I}^u$  and  $v_j \in \mathbb{R}^{k \times 1}$  represent a candidate item  $i_j^u$ . Hence the matrix  $C$  characterizes the user’s interests and  $k$  is the dimensionality of the latent factor. Although  $C$  stores all item information of user  $u$ , our goal is to learn a deep representation of user  $u$  through the adaptive attention. Since items in  $\mathcal{I}^u$  are diverse and only a subset of  $\mathcal{I}^u$  is relevant to candidate item  $i_j^u$ , we introduce an item-level attention mechanism named as adaptive attention network on the memory component  $C$  to capture items in  $\mathcal{I}^u$  relevant to item  $i_j^u$ . More items in  $\mathcal{I}^u$  with high relevance scores to the item  $i_j^u$  denotes that the user  $u$  is more likely interested in item  $i_j^u$ . The adaptive attention network in the input level which is used to measure the relevance scores between  $i_j^u$  and each item representation in  $C$  is defined as:

$$\alpha_c^u = \frac{\exp(C^T v_j)}{\sum_{i_m \in \mathcal{I}/\mathcal{I}^u} \exp(C^T v_m)} \quad (5)$$

Defined in this way,  $\alpha_c^u \in \mathbb{R}^{n \times 1}$  is the adaptive attention column vector of user  $u$  for item  $i_j^u$  in the input level of GRU. The larger value in  $\alpha_c^u$ , the higher relevance between item  $i_j^u$  and a corresponding item in  $\mathcal{I}^u$  and a higher weight for deriving interests of user  $u$  for item  $i_j^u$ . Therefore, the input of GRU is formed as:

$$V^u = \alpha_c^u \odot C \quad (6)$$

where  $\odot$  is the element-wise product and  $V^u = \{v_1^u, v_2^u, \dots, v_n^u\}$ . The formulas of GRU will be rewritten as:

$$r_t = \sigma(v_t^u W_{xr} + h_{t-1} W_{hr} + b_r) \quad (7)$$

$$z_t = \sigma(v_t^u W_{xz} + h_{t-1} W_{hz} + b_z) \quad (8)$$

$$\tilde{h}_t = \tanh(v_t^u W_{xh} + r_t \odot h_{t-1} W_{hh} + b_h) \quad (9)$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t \quad (10)$$

Let  $H = \{h_1, h_2, \dots, h_n\}$  represent the set of intermediate output of GRU and  $v_t^u$  denote the item in  $V^u$  at time step  $t$ . In this way, the  $h_t$  contains the information of relevance of candidate item  $i_j^u$  and the previous input items.

**Adaptive Attention Network in Hidden State Level.** The hidden state level is to further relieve the monotonic assumption problem and give different weights to current hidden state, especially the final hidden state. Since the RNN-based methods mostly employ the final hidden state as the user’s representation

then making recommendations to users, we use the adaptive attention again to get a deep adaptive user representation.

Similar to the adaptive attention in the input level,  $H$  characterizes the user's representation at each time step. With the  $H$  and  $v_j$ , we measure the relevance between each hidden state in  $H$  and item  $i_j^u$ . Thus, the adaptive attention network in hidden state level is formed as:

$$\alpha_h^u = \frac{\exp(H^T v_j)}{\sum_{i_m \in \mathcal{I}/\mathcal{I}^u} \exp(H^T v_m)} \quad (11)$$

The larger value in  $\alpha_h^u$ , the higher relevance between item  $i_j^u$  and corresponding hidden state in  $H$ . Consider that GRU can transmit previous information to the current state, it can enable the final hidden state to focus on the diversified interests of users through giving different weights to the mediate hidden state. The formula of the hidden state set which contains different weights is as follows:

$$H^u = \alpha_h^u \odot H \quad (12)$$

where  $\odot$  is the element-wise product and the final hidden state with attention is the adaptive user representation which is adaptively focusing on items in  $\mathcal{I}^u$  activated by item  $i_j^u$ .

**Output Level.** In the output level, let  $h_u^j$  denote the final hidden state with attention of GRU. We use  $h_u^j$  to represent the deep adaptive user representation. Recall that the adaptive user representation  $h_u^j$  is obtained by placing the adaptive attention on the memory component  $C$  and the set of hidden state  $H$ . Given the representation of a candidate item  $i_j^u$ , we can compute the user's preference score over item  $i_j^u$  as:

$$\hat{y}_u^j = (h_u^j)^T v_j \quad (13)$$

$\hat{y}_u^j$  is the preference score which reflects the preference of user  $u$  for item  $i_j^u$ .

### 4.3 Network Learning

The goal of our model is to recommend a ranked list of items that satisfy user's interests. To train 3AGRU optimized for ranking inspired by BPR, we formalize the training data  $\mathcal{D}$  by  $(u, p, q)$  triples as:

$$\mathcal{D} = \{(u, p, q) | u \in \mathcal{U} \wedge p \in \mathcal{I}^u \wedge q \in \mathcal{I}/\mathcal{I}^u\} \quad (14)$$

where  $u$  denotes the target user,  $p$  and  $q$  represent the positive and negative items respectively. Item  $p$  is from user's history  $\mathcal{I}^u$  while item  $q$  is randomly chosen from the rest items. Similar to BPR, instead of scoring single items, we use item pairs to calculate user's preference for positive and negative items. The objective function minimizes the following formula:

$$\mathcal{L} = \arg \min_{\theta} \sum_{(u,p,q) \in \mathcal{D}} -\ln(\sigma(\hat{y}_u^p - \hat{y}_u^q)) + \frac{\lambda}{2} \|\theta\|^2 \quad (15)$$

where  $\hat{y}_u^p$  and  $\hat{y}_u^q$  represent the user  $u$ 's preference score over item  $p$  and  $q$ .  $\theta$  represents all of the model parameters that are learned while  $\lambda$  is the regularization terms. The sigmoid function  $\sigma$  maps user  $u$ 's preference score of item  $p$  and  $q$  into probabilities.

## 5 Experiment

In this section, we first explain the setup of experiments, and then analyze the results of experiments. After that, we explore the advantages of the adaptive attention network over the fixed attention mechanism. Finally, we present the influence of hyper-parameters. We aim to answer these questions as follows:

- Q1: What's the performance of 3ARGU, comparing to other state-of-the-art methods?  
 Q2: What's the advantage of adaptive attention, comparing to fixed attention?  
 Q3: How do the parameters affect model performance, such as the dimension of latent vectors and the regularization terms?

### 5.1 Experimental Setup

**Datasets.** We conduct the experiments on the three datasets with different kinds of items. The basic statistics of datasets are listed in Table 1. Specifically, CA is a Foursquare<sup>1</sup> dataset where users are in California. It was collected from January 2010 to February 2011, and used in [4]. Gowalla<sup>2</sup> and Brightkite<sup>3</sup> are two widely used LBSN datasets, which contain massive implicit feedbacks through user-POI check-ins. To remove rare cases, we eliminated users' interactions with fewer than 15 items and items interacted by fewer than 10 users in the three datasets.

**Table 1.** Statistics of datasets

Dataset	Users	Items	Feedbacks	Avg.seq.len	Density
CA	2031	3112	106,229	52.30	1.68%
Gowalla	5073	7021	252,944	49.87	0.71%
Brightkite	1850	1672	257,369	139.12	8.12%

**Baseline.** To evaluate the performance of 3ARGU, we compare it with following comparative methods.

<sup>1</sup> <https://sites.google.com/site/yangdingqi>.

<sup>2</sup> <http://snap.stanford.edu/data/loc-gowalla.html>.

<sup>3</sup> <http://snap.stanford.edu/data/loc-brightkite.html>.

- **BPR-MF**: Bayesian Personalized Ranking based Matrix Factorization (BPR-MF) is a popular method for top-N recommendation. It is based on users' pair-wise preferences and neglects the usage of item content.
- **FPMC**<sup>4</sup> [19]: Factorizing Personalized Markov Chain (FPMC) was designed to predict the items in the next basket. It can not only learn the general taste of a user by factorizing the matrix over observed user-item preferences, but also model sequential behaviors by learning a transition graph over items, which is used to predict the next action based on the recent actions of a user.
- **GRU**: Gated Recurrent Unit based RNN (GRU) is the most advanced method for sequential recommendation, which can capture the long-term dependency and compact vanishing gradients of RNN to recommend following items.
- **GRU-ATT** [2]: Similar to our model 3AGRU, we apply the fixed attention mechanism on the input level and hidden state level of GRU, since the fixed attention mechanism [23] has been used in text classification tasks and has a good preference. Here, we name this method GRU-ATT.
- **RUM**<sup>5</sup> [1]: RUM utilizes external memories to improve sequential recommendation, which contains two variants, item-level (RUM<sup>I</sup>) and feature-level (RUM<sup>F</sup>).

**Evaluation Metrics.** For next-future recommendation whose aim is to recommend next item collection that user would probably interact with in the future, we hold out the first 70% of each user's interaction sequence in the time order as the training set and the remaining 30% for testing. Besides, we remove the duplicate items in each user's test sequence. Following previous work [7, 10], we randomly sample 50 items that are not interacted by the user, while other 50 items are samples according to the popularity. We expect that the recommendation system can not only retrieve relevant items out of all available items, but also retrieve the results as accurately as possible. Besides, we hope it can provide a ranking where items of user's interests are ranked in the top. Therefore, we use the Precision, Recall and Mean Average Precision (MAP) to evaluate the preference of our model.

- **Precision, Recall.** The precision is widely used to measure the predictive accuracy in sequential recommendation system area.  $\mathcal{P}@K$  represents the proportion of test cases that recommend items correctly in a top  $K$  position in generated recommendation list for a user. The recall is used to measure how well a model can recommend relevant items out of all available items.  $\mathcal{R}@K$  represents the proportion of test cases that recommend items correctly in a top  $K$  position in the set of user's interacted items in the test data. The definitions of  $\mathcal{P}@K$  and  $\mathcal{R}@K$  are given as:

$$\mathcal{P}@K = \frac{\#Hits}{\text{length of generated recommendation list}} \quad (16)$$

<sup>4</sup> <https://github.com/khesui/FPMC>.

<sup>5</sup> <https://github.com/ch-xu/RUM>.

$$\mathcal{R}@K = \frac{\#Hits}{\text{total number of items the user likes}} \quad (17)$$

- **MAP.** Mean Average Precision is the average of AP to measure the ranking performances.

$$MAP = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} AveP(u), \quad (18)$$

$$AveP(u) = \frac{1}{|K'|} \sum_{k=1}^{K'} p_u(k) rel_u(k) \quad (19)$$

where  $p_u(k)$  represents the precision of the top  $k$  products recommendation to user  $u$ ;  $rel_u(k)$  denotes whether the  $k_{th}$  item has interacted with user  $u$  in the test data;  $K'$  is the cut-off point.

## 5.2 Comparison of Performance

Our experimental results of 3AGRU and the several comparative methods on the three real-world datasets are shown in Table 2. From Table 2, we can observe following phenomena.

First, in terms of the three evaluation metrics (i.e., precision, recall and MAP), 3AGRU consistently outperforms other methods with a large margin on all three real-world datasets. This indicates that 3AGRU is capable to model complicated process of sequential decision through adaptive attention network.

Second, the performance of BPR-MF, which contains no sequential information, is the worst among that of all competitive methods under most cases. This shows that sequential information is important for improving recommendation performance, which confirms that user's interests are dynamic. FPMC, which considers sequential information, has a certain improvement, comparing with BPR-MF. However, since the sequential information considered is based on users' recent actions in FPMC, the effectiveness is not as good as other methods utilizing long-term and short-term information such as GRU.

Third, RUM models use historical records with external memories, and aim to solve the representation of fixed hidden layers and make recommendation more explanatory. From Table 2, we can see that in terms of MAP, RUM on both item-level and feature-level does not perform as well as GRU on the Gowalla dataset, but it performs well on the other two datasets. This is possibly because the data distribution of the Gowalla dataset is relatively sparse. This suggests that the use of attention mechanisms in the hidden state level can describe the user's dynamic preferences.

Fourth, Table 2 shows that GRU-ATT performs much better than GRU. This is because GRU-ATT uses the fixed attention network. It is known that the attention mechanism can improve the performance of recommendation to a certain extent. This also proves that it is effectiveness of the attention mechanism to capture users' evolving appetites for items.



**Table 2.** Experimental results of different methods on three real-world datasets. Note that the larger the number in the table, the better the performance is. And the boldface results are the best while the second best are underlined.

Dataset	Method	@5		@10		@15		@20		MAP
		Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	
CA	BPR-MF	0.0282	0.0108	0.0207	0.0164	0.0170	0.0200	0.0153	0.0233	0.0136
	FPMC	0.0430	0.0191	0.0267	0.0235	0.0203	0.0266	0.0170	0.0296	0.0230
	RUM <sup>I</sup>	0.0633	0.0253	0.0470	0.0372	0.0387	0.0454	0.0348	<u>0.0538</u>	<u>0.0371</u>
	RUM <sup>F</sup>	0.0500	0.0206	0.0419	0.0247	0.0370	0.0279	0.0337	0.0308	0.0360
	GRU	0.0746	0.0266	0.0525	0.0363	0.0415	0.0429	0.0349	0.0476	0.0260
	GRU-ATT	<u>0.0845</u>	<u>0.0293</u>	<u>0.0576</u>	<u>0.0398</u>	<u>0.0460</u>	<u>0.0481</u>	<u>0.0382</u>	0.0533	0.0287
	3AGRU	<b>0.1074</b>	<b>0.0502</b>	<b>0.0782</b>	<b>0.0731</b>	<b>0.0591</b>	<b>0.0828</b>	<b>0.0527</b>	<b>0.0986</b>	<b>0.0477</b>
Gowalla	BPR-MF	0.0105	0.0234	0.0150	0.0172	0.0183	0.0144	0.0208	0.0127	0.0111
	FPMC	0.0510	0.0261	0.0329	0.0317	0.0247	0.0364	0.0202	0.0394	0.0243
	RUM <sup>I</sup>	0.0406	0.0218	0.0299	0.0307	0.0245	0.0367	0.0216	0.0425	0.0158
	RUM <sup>F</sup>	0.0236	0.0117	0.0188	0.0184	0.0159	0.0228	0.0146	0.0269	0.0106
	GRU	0.0634	0.0271	0.0412	0.0347	0.0321	0.0397	0.0270	0.0438	0.0252
	GRU-ATT	<u>0.0718</u>	<u>0.0314</u>	<u>0.0445</u>	<u>0.0377</u>	<u>0.0334</u>	<u>0.0418</u>	<u>0.0273</u>	<u>0.0450</u>	<u>0.0267</u>
	3AGRU	<b>0.0756</b>	<b>0.0418</b>	<b>0.0488</b>	<b>0.0512</b>	<b>0.0382</b>	<b>0.0583</b>	<b>0.0338</b>	<b>0.0669</b>	<b>0.0344</b>
Brightkite	BPR-MF	0.0222	0.0044	0.0198	0.0079	0.0173	0.0105	0.0160	0.0128	0.0068
	FPMC	0.0101	0.0066	0.0088	0.0109	0.0080	0.0151	0.0079	0.0191	0.0122
	RUM <sup>I</sup>	0.0728	<u>0.0199</u>	<u>0.0556</u>	<u>0.0320</u>	<u>0.0476</u>	<u>0.0404</u>	<u>0.0423</u>	<u>0.0483</u>	0.0156
	RUM <sup>F</sup>	0.0356	0.0092	0.0309	0.0166	0.0263	0.0202	0.0216	0.0211	<u>0.0240</u>
	GRU	0.0685	0.0120	0.0478	0.0181	0.0383	0.0221	0.0329	0.0256	0.0136
	GRU-ATT	<u>0.0786</u>	0.0139	0.0531	0.0200	0.0418	0.0246	0.0354	0.0285	0.0150
	3AGRU	<b>0.1129</b>	<b>0.0767</b>	<b>0.0769</b>	<b>0.1031</b>	<b>0.0568</b>	<b>0.1137</b>	<b>0.0467</b>	<b>0.1229</b>	<b>0.0680</b>

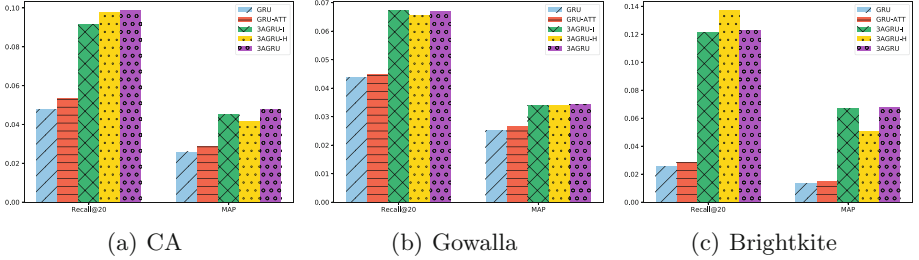
At last, we can make comparisons between our model 3AGRU and GRU-ATT, since both have adopted the attention mechanism. Table 2 shows the advantage of 3AGRU, comparing with GRU-ATT in terms of all three evaluation metrics. This indicates that the adaptive attention network is better than the fixed attention mechanism. We will further explore the advantage of the adaptive attention network over the fixed attention mechanism in the next subsection.

In Summary, our experimental results prove that our proposed model 3AGRU does reflect the user’s interest effectively and performs better than other state-of-the-art methods.

### 5.3 Influence of Components

In order to judge the advantage of the adaptive attention network, comparing to the fixed attention mechanism, we conduct experiments on cases where the adaptive attention network is applied only to the input level or only to the hidden level of GRU. We use 3AGRU-I to present that the adaptive attention network is only applied in the input level, while 3AGRU-H is used to indicate the adaptive attention network is only applied to the hidden state level.

We will observe the results of experiments on 3AGRU-I, 3AGRU-H, 3AGRU and two competitive methods (i.e., GRU and GRU-ATT). Our experimental results are shown in Fig. 2. Due to space limitation, we only show their performance in terms of Recall@20 and MAP.



**Fig. 2.** Exploration of the role of fixed attention mechanism and adaptive attention mechanism in terms of Recall@20 and MAP.

Figure 2 shows that the fixed attention mechanism indeed improves the performance of sequential recommendation to a certain extent. However, we can see that the approaches (i.e., 3AGRU-I, 3AGRU-H, and 3AGRU) based the adaptive attention network perform better than GRU and GRU-ATT. This shows that the adaptive attention network is much better than the fixed attention mechanism. This can prove that the adaptive attention network can better depict users’ dynamic preferences.

From Fig. 2, we can also find that the relationship between 3AGRU and its variants (i.e., 3AGRU-I and 3AGRU-H) is different in different datasets. In terms of performance on CA and Gowalla, 3AGRU performs more or less better than both 3AGRU-I and 3AGRU-H, this indicates that the adaptive attention mechanism is better applied to both the input level and the hidden state level, instead of only one level. This is because the diversity and dynamics of user preferences are further depicted in the hidden state level according to the items to be recommended. In addition, the effect of 3AGRU is only slightly improved compared with that of 3AGRU-I and 3AGRU-H, possibly because users’ preferences for a given item via an adaptive attention network are so accurate that adding another adaptive attention network doesn’t make much difference. When comparing 3AGRU-I and 3AGRU-H, we note that sometimes 3AGRU-I works better and sometimes 3AGRU-H works better, supporting that both the input level and the hidden state level play the important role in GRU. We further investigate the performance on Brightkite. We can observe that 3AGRU-H performs best in terms of Recall@20, this may be operations of GRU for forgetting and updating information play a larger role in dense datasets, making input based on adaptive attention network to some extent forgotten. And the original input to do the same operation, the disadvantages are relatively small.

The above analysis shows that the core part of our model 3AGRU, adaptive attention network, can embody dynamic and diversity characteristics of user sequential preferences. It is indeed more effective than fixed attention mechanism.

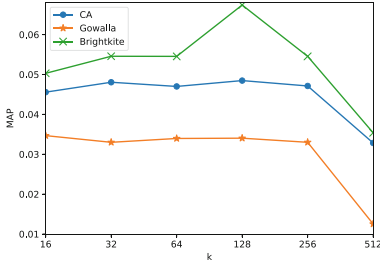


Fig. 3. Influence of dimension size

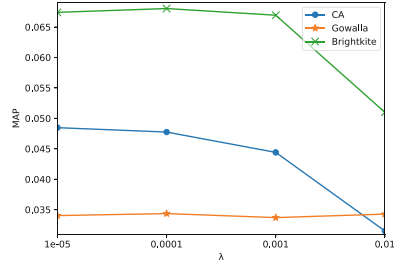


Fig. 4. Influence of regularization terms

### 5.4 Influence of Hyper-parameter

In this subsection, we explore the influences of the dimension size and different regularization terms in our model 3AGRU. Due to space limitation, we just show the results under the metric of MAP.

**Influence of Dimension Size  $k$ .** Dimension size in our model is relevant not only item embedding sizes but also hidden unit size in GRU network which represents the number of features in the hidden state. The size of hidden unit represents the number of nodes that used to remember and store previous states and shows the capability of GRU while dimension size reflects the ability of latent vector representation. Therefore, we choose  $k$  in  $\{16, 32, 64, 128, 256, 512\}$  to find the size which makes the performance of our model 3AGRU better. It can be seen in Fig. 3 that, the performance of the model is improved at the beginning and reaches the best at  $k = 128$  with the increase of  $k$ , and then the performance starts to deteriorate. As Brightkite dataset is relatively dense, this trend is particularly evident in it. In terms of dimension size, it indicates that low-dimension vector has a limitation of modeling complex interactions while the high-dimension vector may affect the generalization of the model and increase the number of parameters. In terms of hidden unit size, proper hidden unit size can help achieve best performance.

**Influence of Different Regularization Terms.** We further investigate the influence of different regularization terms  $\lambda$ . In our model, we utilize  $\mathcal{L}_2$  regularization terms ( $\lambda$ ) mainly on the representation vector of items to avoid overfitting problem. We search the  $\lambda$  from  $\{0.00001, 0.0001, 0.001, 0.01\}$  to optimize performance of our model. Figure 4 shows the influence of regularization at MAP. As shown in the figure, small  $\lambda$  can improve our model in terms of MAP and the 3AGRU reaches its best preference when  $\lambda$  is set to 0.0001. When the value of  $\lambda$  continues to decrease, the performance hardly changes.

## 6 Conclusion

In this paper, we proposed a novel model named Adaptive Attention-Aware Gated Recurrent Unit (3AGRU) to learn adaptive user sequential representations and capture users' short-term and long-term interests to embody the diversity and the dynamics of user interests. With the items to be recommended and users' history records, we constructed a novel attention mechanism called adaptive attention mechanism to reflect the diverse interests of users. First, we embedded the sequence of inputs and the targets into low-rank dimension spaces, and then generated the adaptive attention network in the input level and the hidden state level to adapt the representation of user sequential preferences, and to learn the interactions between steps and items from data. Experimental results demonstrated that 3AGRU achieved a good performance in terms of precision, recall, and MAP, comparing with several competitive methods on the three real-world datasets.

**Acknowledgement.** This research was partially supported by the NSFC (61876117, 61876217, 61872258, 61728205), the Suzhou Science and Technology Development Program (SYG2 01803) and the Open Program of Neusoft Corporation (SKLSAOP1801).

## References

1. Chen, X., et al.: Sequential recommendation with user memory networks. In: WSDM, pp. 108–116. ACM (2018)
2. Cui, Q., Wu, S., Huang, Y., Wang, L.: A hierarchical contextual attention-based GRU network for sequential recommendation. arXiv preprint [arXiv:1711.05114](https://arxiv.org/abs/1711.05114) (2017)
3. Devooght, R., Bersini, H.: Long and short-term recommendations with recurrent neural networks. In: Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization, pp. 13–21. ACM (2017)
4. Gao, H., Tang, J., Liu, H.: gSCorr: modeling geo-social correlations for new check-ins on location-based social networks. In: CIKM, pp. 1582–1586. ACM (2012)
5. Grbovic, M., et al.: E-commerce in your inbox: product recommendations at scale. In: SIGKDD, pp. 1809–1818. ACM (2015)
6. He, R., Kang, W.C., McAuley, J.: Translation-based recommendation. In: RecSys, pp. 161–169. ACM (2017)
7. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: WWW, pp. 173–182. IW3C2 (2017)
8. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based recommendations with recurrent neural networks. In: ICLR (2016)
9. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: 2008 Eighth IEEE International Conference on Data Mining, ICDM 2008, pp. 263–272. IEEE (2008)
10. Huang, J., Zhao, W.X., Dou, H., Wen, J.R., Chang, E.Y.: Improving sequential recommendation with knowledge-enhanced memory networks. In: SIGIR, pp. 505–514. ACM (2018)
11. Lee, J.S., Jun, C.H., Lee, J., Kim, S.: Classification-based collaborative filtering using market basket data. *Expert Syst. Appl.* **29**(3), 700–704 (2005)

12. Li, J., Ren, P., Chen, Z., Ren, Z., Lian, T., Ma, J.: Neural attentive session-based recommendation. In: CIKM, pp. 1419–1428. ACM (2017)
13. Li, Z., Zhao, H., Liu, Q., Huang, Z., Mei, T., Chen, E.: Learning from history and present: next-item recommendation via discriminatively exploiting user behaviors. In: KDD, pp. 1734–1743. ACM (2018)
14. Lin, X., Niu, S., Wang, Y., Li, Y.: K-plet recurrent neural networks for sequential recommendation. In: SIGIR, pp. 1057–1060. ACM (2018)
15. Linden, G., Smith, B., York, J.: Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet comput.* **1**, 76–80 (2003)
16. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: *Advances in Neural Information Processing Systems*, pp. 3111–3119 (2013)
17. Pan, R., Scholz, M.: Mind the gaps: weighting the unknown in large-scale one-class collaborative filtering. In: SIGKDD, pp. 667–676. ACM (2009)
18. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: *Proceedings of the Twenty-fifth Conference on Uncertainty in Artificial Intelligence*, pp. 452–461. AUAI Press (2009)
19. Rendle, S., Freudenthaler, C., Schmidt-Thieme, L.: Factorizing personalized Markov chains for next-basket recommendation. In: WWW, pp. 811–820. ACM (2010)
20. Shani, G., Heckerman, D., Brafman, R.I.: An MDP-based recommender system. *J. Mach. Learn. Res.* **6**(Sep), 1265–1295 (2005)
21. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. *Adv. Artif. Intell.* **2009**, 19 (2009)
22. Wang, P., Guo, J., Lan, Y., Xu, J., Wan, S., Cheng, X.: Learning hierarchical representation model for nextbasket recommendation. In: SIGIR, pp. 403–412. ACM (2015)
23. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E.: Hierarchical attention networks for document classification. *NAACL-HLT* **2016**, 1480–1489 (2016)
24. Ying, H., et al.: Sequential recommender system based on hierarchical attention networks. In: IJCAI (2018)
25. Yu, F., Liu, Q., Wu, S., Wang, L., Tan, T.: A dynamic recurrent model for next basket recommendation. In: SIGIR, pp. 729–732. ACM (2016)
26. Zhang, Y., Wang, H., Lian, D., Tsang, I.W., Yin, H., Yang, G.: Discrete ranking-based matrix factorization with self-paced learning. In: SIGKDD, pp. 2758–2767. ACM (2018)
27. Zhang, Y., et al.: Sequential click prediction for sponsored search with recurrent neural networks. In: AAAI, vol. 14, pp. 1369–1375 (2014)



# Attention and Convolution Enhanced Memory Network for Sequential Recommendation

Jian Liu<sup>1</sup>, Pengpeng Zhao<sup>1,5(✉)</sup>, Yanchi Liu<sup>2</sup>, Jiajie Xu<sup>1,5</sup>, Junhua Fang<sup>1</sup>,  
Lei Zhao<sup>1</sup>, Victor S. Sheng<sup>3</sup>, and Zhiming Cui<sup>4</sup>

<sup>1</sup> Institute of Artificial Intelligence, School of Computer Science and Technology,  
Soochow University, Suzhou, China

jliu2013@stu.suda.edu.cn, {ppzhao,xujj,jhfang,zhao1}@suda.edu.cn

<sup>2</sup> Rutgers University, New Brunswick, USA

yanchi.liu@rutgers.edu

<sup>3</sup> University of Central Arkansas, Conway, USA

ssheng@uca.edu

<sup>4</sup> SuZhou University of Science and Technology, Suzhou, China

zmcui@mail.usts.edu.cn

<sup>5</sup> Neusoft Corporation, Shenyang, China

**Abstract.** The sequential recommendation, which models sequential behavioral patterns among users for the recommendation, plays a critical role in recommender systems. Conventionally, user general taste and recent demand are combined to promote recommendation performance. However, existing methods usually neglect that user long-term preference keeps evolving over time and only use a static user embedding to model the general taste. Moreover, they often ignore the feature interactions when modeling short-term sequential patterns and integrate user-item or item-item interactions through a linear way, which limits the capability of model. To this end, we propose an Attention and Convolution enhanced memory network for Sequential Recommendation (ACSR) in this paper. Specifically, an attention layer learns user's general preference, while the convolutional layer searches for feature interactions and sequential patterns to capture user's sequential preference. Moreover, the outputs of the attention layer and the convolutional layer are concatenated and fed into a fully-connected layer to generate the recommendation. This approach provides a unified and flexible network structure for capturing both general taste and sequential preference. Finally, we evaluate our model on two real-world datasets. Extensive experimental results show that our model ACSR outperforms the state-of-the-art approaches.

**Keywords:** Sequential recommendation · Attention mechanism · Neural network

# 1 Introduction

With the fast development of platform economy, many companies like Amazon, TaoBao, and Uber are creating self-ecosystems to retain users through interactions with products and services. Users can access these platforms through mobile devices in daily life. As a result, a great amount of behavior logs have been generated. For instance, 68 million user trips have been accumulated in June 2017 at Uber, and more than 11 billions check-ins have been generated by over 50 million users at Foursquare. To build effective recommender systems, a key factor is to accurately characterize and understand user's interests and tastes, which are intrinsically dynamic and evolving. To achieve this goal, the sequential recommendation has been proposed to recommend successive items that a user is likely to interact with based on the historical activity sequences.

Different from conventional recommender systems, more and more data is originated from transactions or sessions in sequential recommendation. These transactions or sessions form user's sequential patterns, where next few items are more likely depended on the items engaged recently by a user. For instance, buying milk and butter together leads to a higher probability of buying flour than buying milk or butter individually. However, conventional recommendation methods like collaborative filtering [17], matrix factorization [9], and top-N recommendation [14], are not suitable for capturing sequential patterns because they do not model the order of actions. To solve this problem, early sequential methods based on Markov chains, such as [4, 5, 16], usually employ separate models to characterize user's long-term preference (i.e., a general taste) and short-term preference (i.e., sequential preferences), and then integrate them together. However, these Markov chains-based methods model local sequential behaviors between every two adjacent items but have difficulties to model high order relationships. Recently, deep neural networks have been intensively studied in related domains and have a great influence on sequential recommendation. The most popular neural network used to model user's sequential patterns is Recurrent Neural Network (RNN). RNN-based methods become more powerful than traditional sequential methods do. However, RNN assumes that temporal dependency changes monotonically [12], which means that the current or hidden state is more important than the previous one. In the sequential recommendation, not all adjacent actions have dependent relationships. To deal with this problem, Caser [18] adopts convolutional filters from Convolutional Neural Network (CNN) and models sequential patterns as local features of the embeddings of previous items. However, this method just treats user embedding as user's general taste that may not be adequate to learn user's general taste which is also very important for sequential recommendation [5, 16].

In this paper, we propose an Attention and Convolution enhanced memory network for Sequential Recommendation, namely ACSR. It leverages both strengths of attention mechanism and CNN to capture user's long-term and short-term preferences and makes a non-linear combination of the long-term preference and short-term preference for recommendation. Specifically, we mine user's general taste by an attention network through user's interaction records

firstly, and then we utilize convolutional filters to search for user’s local sequential features. Horizontal convolutional filters are introduced to capture non-linear feature interactions, while the vertical convolutional filters are used for searching for non-monotone local patterns. And then we concatenate the outputs of the two types of convolutional layers and feed them into a fully-connected neural network layer to get more high-level and abstract features. Finally, the output of the attention network and the output of the convolutional neural network are concatenated together to describe user’s overall interest, and then feed them into another fully-connected neural network layer for generating recommendation. To learn the parameters of our model ACSR, we employ the Bayesian personalized ranking optimization criterion to generate a pair-wise loss function [16]. From our experimental results on two real-world datasets, we can observe that ACSR outperforms state-of-the-art methods. Our contributions in this paper can be summarized as follows.

- We propose a novel sequential recommendation network ACSR, which utilizes attention network and conventional neural network to model general taste and short-term preference, respectively, to generate a high-level hybrid representation of a user.
- We introduce horizontal convolutional filters to capture non-linear feature interactions and vertical convolutional filters to search for local sequential patterns.
- Experimental results on two large real-world datasets reveal that ACSR outperforms state-of-the-art methods.

## 2 Related Work

Markov chains have been introduced by previous work to model user’s individual and sequential information jointly for traditional recommendation. Rendle et al. [16] combined a factorization method with Markov chains. They used the factorization method to mine user’s general taste, and used Markov chains to model user’s sequential patterns. Following this idea, researchers utilized different methods to extract user’s general taste and short-term preference. Chen et al. [1] and Feng et al. [4] used metric embedding to project items into points in a low-dimension Euclidean space for playlist prediction and successive location recommendation. Liang et al. [11] leveraged word embedding to mine information from item-item co-occurrence to improve the performance of the matrix factorization. However, these methods have a limited capability on capturing high-level user-item interactions, since the weights of different components are fixed.

Recently, researchers turn to graphical models and neural networks in recommender systems. Liu et al. [13] proposed a bi-weighted low-rank graph construction model, which integrates user’s interests and sequential preferences with temporal interval assessment. Cheng et al. [3] combined wide linear models with cross-product feature transformations and employed deep neural network to



learn highly nonlinear interactions. However, this model needs feature engineering to design cross features, which can be rarely observed in high sparse data. To deal with this problem, Xiao et al. [21] designed attentional pooling layers to learn a second-order features based on the traditional factorization machine technology. Wu et al. [20] employed recurrent neural network (RNN) to mine dynamic user and item preferences in trajectory data. However, RNN holds an assumption that temporal dependency changes monotonically [13], which means that the current item or the hidden state is more important than previous one. In sequential recommendation, not all adjacent actions have dependent relationships. While Tang and Wang [18] did not model sequential patterns as adjacent actions, they adopted convolutional filters from CNN and modeled sequential pattern as local features of the embeddings of previous items. Beyond that, attentional mechanisms have been recently explored due to the ability of model user's attention. For example, Li et al. [10] proposed a hybrid encoder with an attention mechanism to emphasize user's main purpose. Ying et al. [22] was designed as a hierarchical attention network to capture user's long- and short-term preference in a high-level way. However, this method did not consider feature interactions when modeling sequential patterns. Our proposed approach takes these problem into consideration.

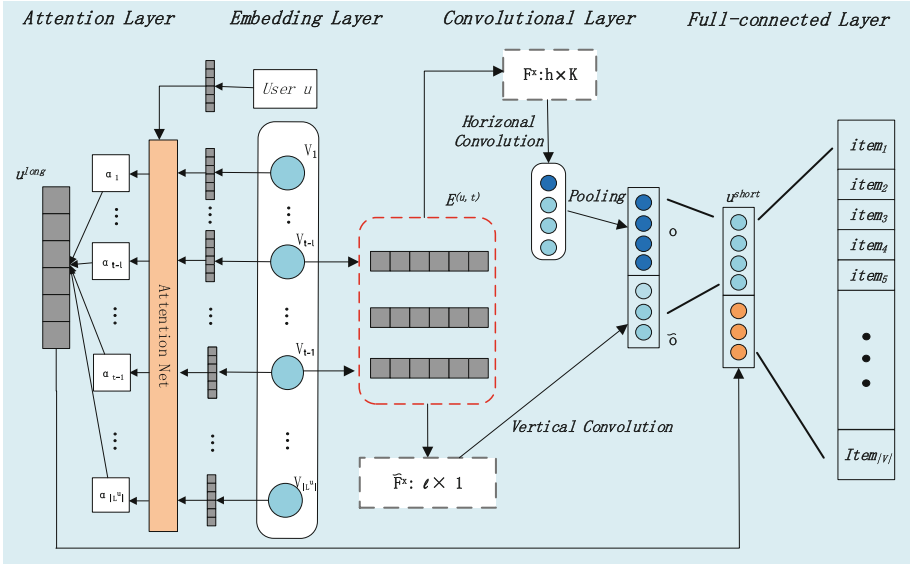
### 3 ACSR Memory Network

In this section, we first describe the problem definition in our work, and then explain the details of our Attention and Convolution Enhanced Memory Network (ACSR). Finally, we will present the optimization procedures of ACSR.

#### 3.1 Problem Definition

Before explaining our model ACSR, we first introduce basic notations that will be used in this paper. Let  $\mathcal{U} = \{u_1, u_2, \dots, u_{|\mathcal{U}|}\}$  denote a set of users, and  $\mathcal{V} = \{v_1, v_2, \dots, v_{|\mathcal{V}|}\}$  denote a set of items, where  $|\mathcal{U}|$  and  $|\mathcal{V}|$  are the number of users and items, respectively. Each user is associated with a sequence of items from  $\mathcal{V}$  and we can form the interaction sequence  $L^u = \{v_1^u, v_2^u, \dots, v_{|L^u|}^u\}$  for each user by sorting interaction records in a chronological order. The index  $t$  for  $v_t^u$  denotes the relative time index, instead of the absolute time index in temporal recommendation like [16, 19, 20].

With these notations, we defined the sequential recommendation task as follows. In this work, we focused on extracting information from implicit and sequential user-item feedback data (e.g., user's successive check-ins and purchase transaction records), which concerns that a user  $u \in \mathcal{U}$  is interacted with an item  $v \in \mathcal{V}$  at time  $t$ . Given a user  $u$  and his/her historical sequence  $L^u$ , the goal is to recommend a list of items that maximize her/his future needs, by considering both general preference and sequential patterns.



**Fig. 1.** The network architecture of ACSR. A hybrid user representation is learned by the attention and convolution neural network, which combines both the general taste and short-term preference.

### 3.2 The Network Architecture of ACSR

As we mentioned before, ACSR incorporates an attention network to learn user's long-term interests and a convolutional neural network to search for users' sequential patterns. As is shown in Fig. 1, ACSR consists of four components, i.e., an Embedding layer, an Attention layer, a Convolutional layer and a Fully-connected Layer. The basic idea of ACSR is to capture a nonlinear high-level representation for each user to generate the recommendation. More specifically, we first embedded sparse user and item inputs (i.e., one-hot representations) into low-dimensional dense vectors, which endows each user or each item an informative representation instead of the basic index. Secondly, we learned users' long-term preference by an attention layer. Then, due to the strength of convolutional filters to capture sequential patterns [18], we searched for local sequential patterns and features interactions by employing horizontal and vertical convolutional filters. Finally, we integrated the output of attention network and the output of convolutional neural network to a fully-connected layer for generating final prediction. The details of each component of ACSR will be explained as follows.

**Embedding Layer.** Similar to discrete word symbols in natural language processing, the original user and item IDs have very limited representation capacity. Therefore, our memory network ACSR first embeds user and item IDs (i.e., one-hot representations) into two continuous low-dimensional spaces. Formally, let

$\mathbf{U} \in \mathbb{R}^{K \times |\mathcal{U}|}$  and  $\mathbf{V} \in \mathbb{R}^{K \times |\mathcal{V}|}$  be two matrices consisting of the user and item embeddings, respectively, where  $K$  is the dimensionality of latent embedding spaces. Theoretically, traditional matrix factorization is equivalent to a two-layer neural network, which constructs low-rank embeddings for users and items in the first layer and employs inner product operation in the second layer. However, embeddings through matrix factorization only capture a low-level, bi-linear and static representation, which limits the capability of representations [11]. On the contrary, our method learns a dynamic and high-level user representation based on these basic embeddings.

**Attention Layer.** In sequential recommender systems, long-term and short-term preferences correspond to user’s general taste and sequential patterns, respectively [16]. Since the long-term item set of a user usually changes over time, learning a static long-term preference representation for each user cannot fully express the dynamics of the user’s long-term preference. Moreover, we argue that the same items might have different impact on different users, and different items have different influences on the next few items that will be purchased. For instance, user  $a$  buys item  $m$  for himself because of interest, while user  $b$  buys item  $m$  as a gift for his/her friend. In this case, it is reasonable to infer that item  $m$  has different weights or attentions on user  $a$  and  $b$  when predicting what they may purchase in the near future.

To meet the above requirements, we turn to use the attention mechanism, which has been successfully applied in many tasks, such as image question answering, document classification and recommendation. Given a user  $u$ , we first compute the importance of each item in his/her history sequence at time step  $t$ , and then we aggregate the embeddings of these items to form the user’s long-term preference representation. Formally, the attention network is defined as follows.

$$\mathbf{h}_j = \phi(\mathbf{W}_1 \mathbf{v}_j + \mathbf{b}_1), \quad (1)$$

$$\alpha_j = \frac{\exp(\mathbf{u}^T \mathbf{h}_j)}{\sum_{m \in L^u} \exp(\mathbf{u}^T \mathbf{h}_m)}, \quad (2)$$

where  $\mathbf{W}_1 \in \mathbb{R}^{K \times K}$  and  $\mathbf{b}_1 \in \mathbb{R}^{K \times 1}$  are model parameters. We assume that the items are consecutively labeled from 1 to  $|\mathcal{V}|$  and  $\mathbf{v}_j$  presents the dense embedding vector of item  $j$ . Firstly, we feed the dense low-dimensional embedding of each item  $j \in \{v_1, \dots, v_{t-1}\}$  through a multi-layer perception (MLP) to get the hidden representation  $\mathbf{h}_j$ .  $\phi(\cdot)$  is an activation function, and RELU is used to enhance nonlinear capability. Unlike traditional attention models that use the same context vectors for each input, we treat the embedding  $\mathbf{u}$  of user  $u$  as the context vector and measure the attention score  $\alpha_j$  as the normalized similarity between  $\mathbf{h}_j$  and  $\mathbf{u}$  with the softmax function, which characterizes the importance of item  $j$  for user  $u$ . Finally, we compute the user’s long-term representation  $\mathbf{u}^{long}$  as a sum of the item embeddings weighted by the attention scores as follows.

$$\mathbf{u}^{long} = \sum_{j \in L^u} \alpha_j \mathbf{v}_j. \quad (3)$$

**Convolutional Layer.** The convolutional layer leverages the recent success of convolutional filters of CNN in capturing local features for image recognition and natural language processing. We present embeddings of the previous  $l$  items (i.e., from  $v_{t-l}$  to  $v_{t-1}$ ) of user  $u$  at time step  $t$  as a  $l \times K$  matrix  $E^{(u,t)}$ , and the rows of the matrix preserve the order of items. And then we regard the matrix  $E^{(u,t)}$  as an “image” to make the horizontal and the vertical convolutional operations respectively. More details are explained below.

**Horizontal Convolutional Layer.** In this layer, we utilize horizontal filters  $\mathbf{F}^x \in \mathbb{R}^{h \times K}$  ( $1 \leq x \leq n$ ,  $h \in [1, \dots, l]$ ) to slide over a sequence and pick up signals for sequential patterns at different time steps, where  $h$  is the height of the horizontal filter,  $n$  is the number of horizontal filters and  $K$  is the size of low-dimension. For each time step  $t$  in the sequence  $L^u$  of user  $u$ , we have a window matrix  $E^{(u,t)} \in \mathbb{R}^{l \times K}$ . Each  $\mathbf{F}^x$  will slide from the top to bottom on  $E^{(u,t)}$  of the item  $i$  ( $1 \leq i \leq l - h + 1$ ). The result of the feature interactions is the  $i$ -th convolution value given by the following equation.

$$\mathbf{c}_i^x = \phi_H(E_{i:i+h-1} \cdot F^x), \quad (4)$$

where  $\phi_H$  is an activation function for convolutional layers, and RELU is used to enhance its nonlinear capability. This value is the inner product between  $F^x$  and the sub-matrix formed by the row  $i$  to  $i + h - 1$  of  $E^{(u,t)}$ , denoted by  $E_{i:i+h-1}$ . The final convolution result of  $F^x$  is the vector as follows.

$$\mathbf{c}^x = [\mathbf{c}_1^x \mathbf{c}_2^x \dots \mathbf{c}_{l-h+1}^x]. \quad (5)$$

Then, we apply a max pooling operation to  $\mathbf{c}^x$  to extract the maximum value from all values produced by this particular filter. The maximum value captures the most significant feature extracted by the filter. Therefore, for the  $n$  filters in this layer, the output value  $\mathbf{o} \in \mathbb{R}^n$  is:

$$\mathbf{o} = \{\max(\mathbf{c}^1), \max(\mathbf{c}^2), \dots, \max(\mathbf{c}^n)\}, \quad (6)$$

Horizontal filters interact with every successive  $h$  items for features interactions through their embeddings  $E^{(u,t)}$ . By sliding the filters of various heights, this layer can search for the feature interactions and item-item interactions.

**Vertical Convolutional Layer.** This layer is shown in the lower part of the second component in Fig. 1. A tilde( $\sim$ ) is used as the symbol of this layer. Suppose that there are  $\tilde{n}$  vertical filters  $\tilde{\mathbf{F}}^x \in \mathbb{R}^{l \times 1}$  for the operation of vertical convolution,  $1 \leq x \leq \tilde{n}$ . For each time step  $t$  in the sequence  $L^u$ , we have a window matrix  $E^{(u,t)}$  with  $l$  previous consecutive items. Each  $\tilde{\mathbf{F}}^x$  interacts with each column of  $E^{(u,t)}$  by sliding  $K$  times from left to right for user  $u$  at time step  $t$  in a valid way. We can obtain the vertical convolutional result  $\tilde{\mathbf{c}}^x \in \mathbb{R}^K$  as follows.

$$\tilde{\mathbf{c}}^x = [\tilde{c}_1^x \tilde{c}_2^x \dots \tilde{c}_K^x]. \quad (7)$$

For the inner product interaction, it is easy to verify that this result is equal to the weighted sum over the  $l$  rows of  $\mathbf{E}^{(u,t)}$  with  $\tilde{\mathbf{F}}^x$  as the weights. That is,

$$\tilde{\mathbf{c}}^x = \sum_{j=1}^l \tilde{\mathbf{F}}_s^x \cdot \mathbf{E}_s^{(u,t)}, \quad (8)$$

where  $s$  is the  $s$ -th row of  $\mathbf{E}^{(u,t)}$ . With the vertical filters, we can learn to aggregate the embeddings of the  $l$  previous items, similar to [5]. Thus, these vertical filters can capture user  $u$ 's local sequential patterns at time step  $t$  through weighted sums over the latent representations of previous items. While [5] used a single weighted sum for each user, we can use  $\tilde{n}$  global vertical filters to produce  $\tilde{n}$  weighted sums  $\tilde{\mathbf{o}} \in \mathbb{R}^{K \times \tilde{n}}$  for all users as follows.

$$\tilde{\mathbf{o}} = [\tilde{\mathbf{c}}^1 \tilde{\mathbf{c}}^2 \dots \tilde{\mathbf{c}}^{\tilde{n}}]. \quad (9)$$

There is no need to apply a pooling operation over the vertical convolutional results, as we want to keep the aggregation for every latent dimensions. Thus, the output of this layer is  $\tilde{\mathbf{o}}$ .

Finally, We concatenate the output of the two convolutional layers and then feed them into a fully-connected neural network layer to get more high-level and abstract features as follows.

$$\mathbf{u}^{short} = \phi_a(\mathbf{W}_2 \begin{bmatrix} \mathbf{o} \\ \tilde{\mathbf{o}} \end{bmatrix} + \mathbf{b}_2), \quad (10)$$

where  $\mathbf{W}_2 \in \mathbb{R}^{K \times (n+K \times \tilde{n})}$  is the weight matrix that projects the concatenation layer to a  $K$ -dimensional hidden layer,  $\mathbf{b}_2$  is the corresponding bias term, and  $\phi_a(\cdot)$  is the activation function RELU for the fully-connected layer.

**Fully-Connected Layer.** To capture user's current overall preference, we also concatenate the output of attention network  $\mathbf{u}^{long}$  and the output of convolutional layer  $\mathbf{u}^{short}$  together and project them into an output layer with  $|\mathcal{V}|$  nodes, denoted as follows.

$$\mathbf{y}^{(u,t)} = \mathbf{W}_3 \begin{bmatrix} \mathbf{u}^{long} \\ \mathbf{u}^{short} \end{bmatrix} + \mathbf{b}_3, \quad (11)$$

where  $\mathbf{b}_3 \in \mathbb{R}^{|\mathcal{V}|}$  and  $\mathbf{W}_3 \in \mathbb{R}^{|\mathcal{V}| \times 2K}$  are the bias term and the weight matrix for the output layer, respectively. Note that the value  $\mathbf{y}_i^{(u,t)}$  in the output layer is associated with the probability of how likely user  $u$  will interact with item  $i$  at time step  $t$ .  $\mathbf{u}^{long}$  captures user long-term preferences, whereas  $\mathbf{u}^{short}$  intends to capture short-term sequential patterns.

### 3.3 Network Learning

The goal of our model is to provide a top-N ranked list of items given the sequence item sets of a user at time  $t$ . We followed the BPR optimization criterion objective function for our model ACSR. We assumed that users prefer the next future purchased items than unobserved items, and defined a ranking order  $\succ_{u,L^u}$  over items  $p$  and  $q$  as follows.

$$p \succ_{u,L^u} p \Leftrightarrow \mathbf{y}_p^{u,t} > \mathbf{y}_q^{u,t}, \quad (12)$$

where  $p$  is the purchased item by user  $u$  at time step  $t$ , and  $q$  is an unobserved item generated by bootstrap sampling. For each observation  $(u, L^u, p)$ , we generated a set of pairwise preference order  $D = \{(u, L^u, p, q)\}$ . Then we trained our model ACSR by maximizing a posterior (MAP) as follows.

$$\arg \min_{\Theta} \sum_{(u, L^u, p, q) \in D} -\ln \sigma(\mathbf{y}_p^{u,t} - \mathbf{y}_q^{u,t}) + \lambda \|\Theta\|^2, \quad (13)$$

where  $\Theta$  presents all of the model parameters that are learned,  $\lambda$  is the regularization weight, and  $\sigma$  is the logistic function.

## 4 Experiment

We have conducted experiments to evaluate our model ACSR on two real-world datasets, by comparing with baseline methods. Before showing our experimental results, we first briefly described the two real-world datasets and our evaluation metrics.

### 4.1 Experimental Setup

**Datasets.** We performed experiments on two real-world datasets, i.e., Gowalla<sup>1</sup> and TaoBao<sup>2</sup>. The dataset Gowalla records the time and the point-of-interest information of check-ins of users in a location-based social networking site, Gowalla. Taobao is the largest B2C platform in China. The dataset TaoBao is from IJCAI 2017 competition. It is a user-purchase dataset, containing users' shop information from July 1, 2015 to October 31, 2016. We removed items that have been observed by less than 10 users and eliminated users interacting with fewer than 15 items. After preprocessing, basic statistics of both datasets are summarized in Table 1.

**Table 1.** Statistics of the datasets

Dataset	#users	#items	#feedbacks	avg.seq.len	sparsity(%)
Gowalla	5,073	7,020	252,944	49.86	99.29
TaoBao	2,499	1,619	76,331	33.24	98.12

**Evaluation Metrics.** In our experiments, we took the first 70% of interaction sequences as the training set, the rest 30% for testing. Like [14], the performance of each method is evaluated on the test set in terms of *Recall@N*, *Precision@N*

<sup>1</sup> <http://snap.stanford.edu/data/loc-gowalla.html>.

<sup>2</sup> <https://tianchi.shuju.aliyun.com/datalab/dataSet>.

and *Mean Average Precision (MAP)*. A list of top  $N$  predicted items for a user is denoted  $\hat{R}_{1:N}$ , where  $R$  presents the corresponding test set. Therefore, *Recall@N* and *Precision@N* are calculated as follows.

$$Recall@N = \frac{|R \cap \hat{R}_{1:N}|}{|\hat{R}_{1:N}|}, \quad (14)$$

$$Pre@N = \frac{|R \cap \hat{R}_{1:N}|}{N}, \quad (15)$$

We conducted each experiment five times, and reported the average performance of all users with  $N \in \{1, 5, 10\}$ . The Average Precision (AP) is define as follows.

$$AP = \frac{\sum_{N=1}^{|\hat{R}|} Pre@N \times rel(N)}{|\hat{R}|}, \quad (16)$$

where  $rel(N) = 1$  if the  $N$ -th item in  $\hat{R}$  is in  $R$ . The Mean average Precision (MAP) is the average of AP for all users. Note that a larger metric value indicates a better performance.

**Baselines.** We compared our model ACSR with baseline methods as follows.

- **BPR-MF** [15] This method is a state-of-the-art framework for evaluating implicit user feedback data through pairwise learning, and matrix factorization is chosen as its internal predictor.
- **FPMC** [16] This method models user preference through matrix factorization and sequential information through first-order Markov chain simultaneously, and then combines them by a linear way for next basket recommendation.
- **FOSSIL** [6] This method integrates factored item similarity with Markov chain to model user's long-term and short-term preferences.
- **LSTM** [7] This method is a variant of RNN, which contains a memory cell and three multiplicative gates to hold long-term dependencies.
- **NARM** [10] This method is a RNN-based state-of-the-art model, which employs an item-level attention mechanism to capture user's main purpose from hidden states and combines it with sequential behaviors as user's final representation to generate recommendation.
- **RUM** [2] This method utilizes external memories to improve sequential recommendation, which contains two variants, i.e., item-level and feature-level variant.
- **Caser** [18] This method is the first model of leveraging convolutional neural network to capture user's sequential patterns.
- **SHAN** [22] This method is a hierarchical attention network, which utilizes two same attention networks to capture long-term and short-term preferences, respectively.

**Parameter Settings.** All methods have several parameters to tune. We either followed the reported optimal parameter settings or optimized each model separately using the validation set. The weight parameters in our model ACSR are randomly initialized by a normal distribution. For other parameters, we searched for their optimal settings. That is, we searched the optimal hidden size  $K$  from  $\{5, 10, 20, 30, 50, 100, 120\}$ , the length of previous items  $l$  from  $\{2, 4, 6, 8, 10, 12\}$ , the number of horizontal filters  $n$  and vertical filters  $\tilde{n}$  from  $\{2, 4, 8, 16, 32\}$ , and the regularization parameter  $\lambda$  from  $\{10^{-2}, 10^{-3}, 10^{-4}\}$ . To control the model complexity and avoid overfitting, we used a dropout technique with 50% drop ratio on the final fully-connected layer.

**Table 2.** The performance of ACSR and the baselines on the two real-world datasets. The best performance is highlighted in boldface (the higher the better).

Datasets	Methods	Pre@1	Pre@5	Pre@10	Rec@1	Rec@5	Rec@10	MAP
Gowalla	BPR-MF	0.1526	0.0787	0.0539	0.0113	0.0320	0.0450	0.0302
	FPMC	0.1531	0.0789	0.0564	0.0126	0.0342	0.0450	0.0315
	FOSSIL	0.1539	0.0801	0.0577	0.0131	0.0354	0.0478	0.0329
	RUM <sup>I</sup>	0.1547	0.0846	0.0597	0.0139	0.0380	0.0512	0.0380
	RUM <sup>F</sup>	0.1570	0.0799	0.0565	0.0149	0.0347	0.0480	0.0360
	LSTM	0.1491	0.0884	0.0652	0.0139	0.0405	0.0600	0.0437
	Caser	0.1679	0.0906	0.0672	0.0164	0.0455	0.0699	0.0519
	NARM	0.1892	0.1034	0.0765	0.0184	0.0506	0.0730	0.0552
	SHAN	0.2053	0.1141	0.0887	0.0189	0.0498	0.0750	0.0621
	ACSR	<b>0.2230</b>	<b>0.1233</b>	<b>0.0951</b>	<b>0.0207</b>	<b>0.0573</b>	<b>0.0815</b>	<b>0.0682</b>
TaoBao	BPR-MF	0.1608	0.1309	0.1052	0.0170	0.0735	0.1207	0.0807
	FPMC	0.2128	0.1416	0.1020	0.0235	0.0764	0.1131	0.0809
	FOSSIL	0.2263	0.1637	0.1273	0.0256	0.0778	0.1148	0.0811
	RUM <sup>I</sup>	0.2313	0.1382	0.1313	0.0248	0.0786	0.1238	0.0842
	RUM <sup>F</sup>	0.2341	0.1487	0.1387	0.0245	0.0849	0.1288	0.0866
	LSTM	0.3973	0.2123	0.1405	0.0491	0.1191	0.1603	0.1246
	Caser	0.4105	0.2195	0.1446	0.0502	0.1231	0.1622	0.1251
	NARM	0.4362	0.2245	0.1489	0.0492	0.1233	0.1695	0.1330
	SHAN	0.4471	0.2419	0.1535	0.0515	0.1409	0.1721	0.1511
	ACSR	<b>0.4588</b>	<b>0.2531</b>	<b>0.1577</b>	<b>0.0592</b>	<b>0.1503</b>	<b>0.1795</b>	<b>0.1632</b>

## 4.2 Comparison of Performance

Table 2 shows the performance of all methods in terms of Recall, Precision and MAP. From this table, we can observe that:

First, BPR-MF has the worst performance under almost all cases among all baselines. This demonstrates that sequential information is very important



for sequential recommendation. The classic model FPMC performs better than BPR-MF under most cases because of taking the sequential correlation between adjacent actions into account. Over all, the experimental results in Table 2 verify that sequential information can help improve the recommendation performance in real-world system.

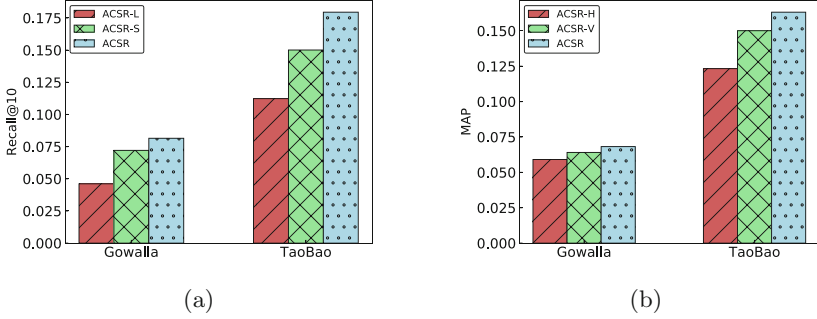
Second, both RNN-based models (i.e., LSTM, NARM) and memory-based models (i.e., Caser, RUM) improve the effectiveness by using deep neural networks. This shows the ability of neural networks on modeling user general taste and sequential behaviors. Caser achieves better performances than LSTM on both datasets. This indicates that Caser can capture more sequential local patterns, which motivates us to combine convolutional neural network with other networks for improvement.

Third, between the two memory-based methods, Caser performs better than RUM on both datasets, especially on TaoBao. This phenomenon indicates that the TaoBao dataset has more sequential signals than the Gowalla dataset, because Caser performs much more better in a dataset with more sequential signals. Over all, both RUM and Caser perform well. This indicates that both KV-MN architecture and convolutional operations can capture user’s sequential preferences. Furthermore, the state-of-the-art attention-based model SHAN beats all the other baselines on both datasets. This shows us that the effectiveness of the attention mechanism to capture user’s appetite for items. SHAN incorporates user’s embedding as extra context into a hierarchical attention network, which may make a greater improvement for recommendation.

Finally, our model ACSR consistently outperforms all the baselines under all measurements on both datasets. For instance, ACSR improves 9.5% and 15% at *Recall@1*, compared with the second best method SHAN, on the Gowalla and the TaoBao dataset, respectively. This indicates that ACSR captures more high-level complicated nonlinear information for long-term and short-term representation through attention and convolutional network, respectively, while SHAN ignores feature interactions that is very important for modeling sequential patterns especially with sparse data [8, 23]. The performance of ACSR is better than Caser, possibly because Caser is not adequate to model the general taste by learning user embedding. Finally, because of the sparsity differences, we also observed that the performances of most models degrade when datasets become sparser.

### 4.3 Influence of Components

To evaluate the contribution of each component for final user preference representation, we conducted experiments of two different variants (i.e., ACSR-L and ACSR-S) of ACSR. ACSR-L means that only user’s general taste is modeled, and ACSR-S means that only user’s sequential preference is considered. Our experimental results are shown in Fig. 2(a). From Fig. 2(a) and Table 2, we can see that ACSR-L achieves better performance than BPR does. Note that BPR also only models long-term preference. Its performance in terms of *Recall@10* is 0.0450 and 0.1207 on the Gowalla and TaoBao dataset, respectively. This indicates that modeling the general taste is better than using an embedding vector. Besides,



**Fig. 2.** (a) shows the influence of long-term and short-term preference in terms of Recall@10. (b) shows the influence of convolutional filters in terms of MAP.

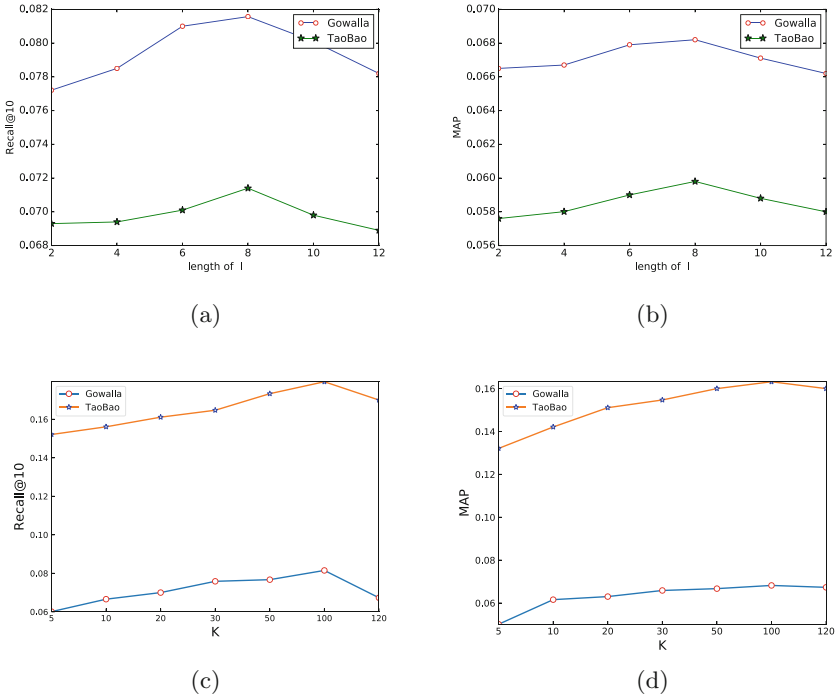
Fig. 2(a) shows that ACSR-S performs better than ACSR-L with a large margin. This demonstrates that short-term sequential information plays a more important role in sequential recommendation. Finally, ACSR performs better than each variant on both datasets. It demonstrates that considering both long-term and short-term preferences is important to improve sequential recommendation. Moreover, we also evaluated the contribution of the horizontal and the vertical filters on both datasets in terms of *MAP*. Our experimental results are shown in Fig. 2(b). ACSR-V is a variant of ACSR that only contains the general taste and its vertical convolutional operations, while ACSR-H is another variant of ACSR that only considers the general taste and its horizontal convolutional operation. From Fig. 2(b), we can see that both ACSR-H and ACSR-V perform worse than ACSR on the two datasets. This indicates that the horizontal and vertical convolutional operations of ACSR can help improve the prediction by considering feature interactions and searching for local sequential patterns, respectively.

#### 4.4 Influence of Hyper-parameters

We also investigated the influence of the parameters, such as the length of previous items  $l$ , the number of dimensions  $K$ , the regularization parameter  $\lambda$ , and the number of convolutional filters  $n$  and  $\tilde{n}$ . Due to space limitation, we just showed our experimental results in terms of three metrics, i.e., *MAP*, *Prec@10* and *Recall@10*.

Figure 3(a) and (b) show the experimental results with different lengths of previous items  $l$  used in the convolutional layer. From Fig. 3(a) and (b), we can see that ACSR achieves the better performance by setting a proper larger length  $l$  on the two datasets, suggesting that the convolutional filters can utilize more extra information provided by a proper larger  $l$ .

Figure 3(c) and (d) show the experimental results with different dimension sizes  $K$  in terms of *Recall@10* and *MAP*. As we mentioned before, the dimension size is relevant to not only the embedding size users and items, but also the parameters of fully-connected layer. From Fig. 3(c) and (d), we can observe that



**Fig. 3.** (a) and (b) show the influence of the length  $l$  on both datasets in terms of Recall@10 and MAP respectively. (c) and (d) show the impact of the dimension size  $K$  on both datasets in terms of Recall@10 and MAP respectively.

high dimensions can embed better for users and items, and are more helpful to build high-level factor interactions through our model ACSR. According to our experimental results, the dimension size can be set as 100 for recommendation quality on both datasets.

Because of the limitation of the space, the experimental results of different values of the regularization parameter are not show in this paper, but we can learn that the regularization parameter has a great impact on the performance of ACSR and our method achieves the best performance when the regularization parameter  $\lambda$  is set to 0.001.

We also conducted experiments to investigate the impact of the number of convolutional filters  $n$  and  $\tilde{n}$ . We applied a grid search over combinations of  $n \in \{2, 4, 8, 16, 32\}$  and  $\tilde{n} \in \{2, 4, 8, 16, 32\}$  for a better recommendation. The experimental results in terms of MAP are shown in Table 3. From Table 3, we can see that ACSR achieves the best performance with setting  $n = 16, \tilde{n} = 8$  on the Gowalla dataset, while it achieves the best performance with setting  $n = 8, \tilde{n} = 4$  on the TaoBao dataset. This may be related to the average sequence length in the two datasets, and the longer sequence needs more filters to search for local sequential patterns.

**Table 3.** The performance of ACSR with varying  $n$  and  $\tilde{n}$  in terms of MAP on Gowalla and Taobao dataset.

Dataset	MAP	$\tilde{n}$				
	$n \backslash \tilde{n}$	$\tilde{n} = 2$	$\tilde{n} = 4$	$\tilde{n} = 8$	$\tilde{n} = 16$	$\tilde{n} = 32$
Gowalla	$n = 2$	0.0597	0.0606	0.0641	0.0616	0.0594
	$n = 4$	0.0617	0.0629	0.0657	0.0640	0.0610
	$n = 8$	0.0634	0.0647	0.0664	0.0655	0.0639
	$n = 16$	0.0658	0.0658	<b>0.0682</b>	0.0669	0.0654
	$n = 32$	0.0641	0.0654	0.0668	0.0654	0.0648
Taobao	$n = 2$	0.1556	0.1611	0.1618	0.1605	0.1621
	$n = 4$	0.1603	0.1628	0.1607	0.1582	0.1612
	$n = 8$	0.1587	<b>0.1632</b>	0.1626	0.1595	0.1609
	$n = 16$	0.1566	0.1623	0.1613	0.1594	0.1582
	$n = 32$	0.1598	0.1615	0.1608	0.1592	0.1575

## 5 Conclusion

In this paper, we propose an Attention and Convolution enhanced memory network for Sequential Recommendation (ACSR), which leverages the strengths of both attention mechanism and convolutional filters for sequential recommendation. Our memory network first utilizes an attention network to capture user’s long-term preference. Then, we use vertical convolutional filters and horizontal convolutional filters to search for non-linear feature interactions and non-monotone local patterns to capture user’s short-term preference. Finally, the outputs of the two networks are concatenated and fed into a fully-connected layer to generate the recommendation. Extensive experimental results on two real-world datasets show that ACSR outperforms the state-of-the-art methods on sequential recommendation.

**Acknowledgement.** This research was partially supported by the NSFC (61876117, 61876217, 618722 58, 61728205), the Suzhou Science and Technology Development Program (SYG2 01803) and the Open Program of Neusoft Corporation (SKLSAOP1801).

## References

1. Chen, S., Moore, J.L., Turnbull, D., Joachims, T.: Playlist prediction via metric embedding. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 714–722 (2012)
2. Chen, X.: Sequential recommendation with user memory networks. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, pp. 108–116 (2018)
3. Cheng, H.T., et al.: Wide & deep learning for recommender systems. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, pp. 7–10 (2016)
4. Feng, S., Li, X., Zeng, Y., Cong, G., Chee, Y.M., Yuan, Q.: Personalized ranking metric embedding for next new poi recommendation. In: IJCAI (2015)

5. He, R., McAuley, J.: Fusing similarity models with Markov chains for sparse sequential recommendation. In: 2016 IEEE 16th International Conference on Data Mining (ICDM), pp. 191–200 (2016)
6. He, R., McAuley, J.: Fusing similarity models with Markov chains for sparse sequential recommendation. In: ICDM, pp. 191–200 (2016)
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.* **9**, 1735–1780 (1997)
8. Hu, L., Cao, L., Wang, S., Xu, G., Cao, J., Gu, Z.: Diversifying personalized recommendation with user-session context. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence, pp. 1858–1864 (2017)
9. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. *Computer* **42**, 30–37 (2009)
10. Li, J., Ren, P., Chen, Z., Ren, Z., Lian, T., Ma, J.: Neural attentive session-based recommendation. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 1419–1428 (2017)
11. Liang, D., Altposaar, J., Charlin, L., Blei, D.M.: Factorization meets the item embedding: regularizing matrix factorization with item co-occurrence. In: Proceedings of the 10th ACM Conference on Recommender Systems, pp. 59–66 (2016)
12. Liu, Q., Wu, S., Wang, L.: Multi-behavioral sequential prediction with recurrent log-bilinear model. *IEEE Trans. Knowl. Data Eng.* **29**, 1254–1267 (2017)
13. Liu, Y., Liu, C., Liu, B., Qu, M., Xiong, H.: Unified point-of-interest recommendation with temporal interval assessment. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1015–1024
14. Pan, R., et al.: One-class collaborative filtering. In: Eighth IEEE International Conference on Data Mining, ICDM 2008, pp. 502–511 (2008)
15. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, 18–21 June 2009, pp. 452–461 (2009)
16. Rendle, S., Freudenthaler, C., Schmidt-Thieme, L.: Factorizing personalized Markov chains for next-basket recommendation, pp. 811–820. ACM (2010)
17. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International Conference on World Wide Web, pp. 285–295 (2001)
18. Tang, J., Wang, K.: Personalized top-n sequential recommendation via convolutional sequence embedding. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, pp. 565–573 (2018)
19. Wang, W., Yin, H., Sadiq, S., Chen, L., Xie, M., Zhou, X.: SPORE: a sequential personalized spatial item recommender system. In: 2016 IEEE 32nd International Conference on Data Engineering (ICDE), pp. 954–965. IEEE (2016)
20. Wu, C.Y., Ahmed, A., Beutel, A., Smola, A.J., Jing, H.: Recurrent recommender networks. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, pp. 495–503 (2017)
21. Xiao, J., Ye, H., He, X., Zhang, H., Wu, F., Chua, T.S.: Attentional factorization machines: learning the weight of feature interactions via attention networks (2017)

22. Ying, H., et al.: Sequential recommender system based on hierarchical attention networks. In: The 27th International Joint Conference on Artificial Intelligence (2018)
23. Zhang, F., Yuan, N.J., Lian, D., Xie, X., Ma, W.Y.: Collaborative knowledge base embedding for recommender systems. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 353–362 (2016)



# Attention-Based Neural Tag Recommendation

Jiahao Yuan, Yuanyuan Jin, Wenyan Liu, and Xiaoling Wang<sup>(✉)</sup>

Shanghai Key Laboratory of Trustworthy Computing,  
East China Normal University, 3663 North Zhongshan Road, Shanghai, China  
{jhyuan,yyj,wyliu}@stu.ecnu.edu.cn, xlwang@sei.ecnu.edu.cn

**Abstract.** Personalized tag recommender systems suggest tags to users when annotating specific items. Usually, recommender systems need to take both users' preference and items' features into account. Existing methods like latent factor models based on tensor factorization use low-dimensional dense vectors to represent latent features of users, items and tags. The problem with these models is using the static representation for the user, which neglects that users' preference keeps evolving over time. Other methods based on base-level learning (*BLL*) only use a simple time-decay function to weight users' preference. In this paper, we propose a personalized tag recommender system based on neural networks and attention mechanism. This approach utilizes the multi-layer perceptron to model the non-linearities of interactions among users, items and tags. Also, an attention network is introduced to capture the complex pattern of the user's tagging sequence. Extensive experiments on two real-world datasets show that the proposed model outperforms the state-of-the-art tag recommendation method.

**Keywords:** Tag recommendation · Attention mechanism · Neural networks

## 1 Introduction

In current web services, users are both consumers and generators of web contents. They have generated a large amount of personalized information. Tag, which is the keyword that users use to annotate web resources such as music, bookmarks and movies, is one of important user-generated contents in web services. A tag often represents a specific characteristic of the related resource, which not only helps users manage their resources, but also helps the website integrate resources and provide personalized content services.

Tag systems can greatly improve the search efficiency of web resources, help websites describe the attributes of users and products, improve the accuracy of recommendations, and alleviate the cold start problem. However, a resource may have a large number of associated tags and users are often unable to think of the appropriate keyword at the first time. Personalized tag recommenders

suggest the user a list of tags that he is likely to use for the specific resource by analyzing the user’s historical tagging behavior. It helps websites increase the chances of getting a resource annotated, remind a user what the resource is about and consolidate the vocabulary across users [11].

The personalized tag recommender system mainly focuses on recommending a personalized set of tags based on the user and the item. Different users may use different tags for the same item since their personal preferences. Inspired by the application of the matrix factorization, tensor factorization techniques, as the generalization of matrix factorization, are used in some works [20, 21, 23]. These models formulate the tag recommendation as the tensor completion task, and learn latent representations for users, items and tags. However, these methods often ignore the information of temporal dynamics since the user’s behaviors have changed with time [3, 12, 27]. As a result, some studies [13, 16, 29] have proposed the base-level learning equation to mimics the way humans draw on items in their long-term memory. Essentially, BLL-like models are based on the time decay and the frequency of tagging. It tends to recommend most recently used tags to users, but this kind of approaches cannot capture the complex behavior patterns of users due to only considering simple factors. Recently, the personalized time-aware tag recommendation has been proposed by considering both personalization and the temporal factor [24], and achieves a better recommendation result.

All of the above methods do not take into account the sequence information in the users’ tagging behavior. In this paper, we utilize the attention-based multi-layer neural networks for learning the complex pattern of the user’s tagging sequence. Tags used by a user in sequence may have strong relevance in a specific topic, e.g., a user may like using the name of stars to annotate movies, but the naive model can’t capture it. Recently, deep neural networks have been widely used in recommender systems [7–9, 27]. Some studies have shown that deep neural network can better model user-item interaction in item recommender systems [7, 8]. Besides, the attention mechanism, which is introduced to provide the ability to reference specific records in neural networks, has been applied in many tasks, such as the neural machine translation [1], reading comprehension [17] and other fields [26, 28]. Nevertheless, there is relatively little work on employing attention model for tag recommendation in contrast to a large amount of literature on general item recommender systems.

This paper proposes an attention-based neural networks framework to address the aforementioned problems. Specifically, we first embed users, items and tags into low-dimensional dense spaces. Just like the tensor factorization technique, tags will be embedded as two vectors for users and items, respectively. Then this paper focuses on the users’ tagging behavior, and an attention layer is employed to compute different weights of tags in the user tagging sequence to generate user-tag representation. Finally, a multi-layer neural network is used to capture user-tag and item-tag interactions. To learn the parameters, we employ the Bayesian personalized ranking optimization criterion to generate a pair-wise loss function [21].



The main contributions of this paper are:

- We present a neural network framework to model latent features of users, items and tags, which is used to capture pairwise nonlinear interactions among users, items and tags.
- We introduce the attention mechanism to model personal preference and to explore the sequence pattern of the users' tagging behavior. To the best of our knowledge, we are the first to use the attention mechanism in tag recommender systems.
- We conduct several experiments on two real-world datasets to demonstrate the performance of our model. The experimental results show that the proposed model outperforms the state-of-art methods.

## 2 Relate Work

### 2.1 Personal Tag Recommendation

Early literature on personalized tag recommendation focuses on collaborative filtering method [18]. The representative works are FlokRank [10,11] and PITF [21]. FolkRank is an adaption of PageRank, which combines PageRank algorithm and the similarity between users. PITF is the pairwise interaction tensor factorization. It is a special case of tensor factorization technique, which is widely used for the tag recommendation. This technique explicitly models the pairwise interactions between users, items and tags and is trained with an adaption of the Bayesian personalized ranking (BPR) criterion. The literature [23] utilizes the High-Order-Singular-Value-Decomposition (HOSVD), which learns latent vectors of users, items, and tags by Tucker Decomposition (TD). Then a better TD model RTF-TD is introduced in [20], which learns parameters by maximizing the ranking measure AUC (area under the ROC-curve). To learn better representations, Gaussian radial basis function is used in [4] to increase the model's capacity based on tensor factorization.

Since methods mentioned above all ignore that users' tagging behaviors change over time, some studies explore temporal dynamics in tag recommendation. Based on the frequency and the time information, the work [29] proposes GIPR model and extends it by considering the most popular tags. Then exponential distribution is applied to model the interval of tagging behaviors.  $BLL_{AC}$  [15],  $BLL + MP_m$  [16] and  $BLL_{ac} + MP_m$  [13] are proposed based on the theory of human memory. Essentially, these methods tend to recommend tags which are used recently and consider the frequency or association component to capture features of items. However, these methods only recommend tags that the user has used and cannot capture complex features of users, items, and tags. In [22], the topic model LDA is applied to process the semantic information, which is to mimic the human category learning for better representations of users and items.

Recently, the method TAPITF has been proposed in [24]. It extends PITF by adding weights to user-tag interaction and item-tag interaction respectively

and calculates the weight using the idea from *BLL*. The work has shown that introducing temporal information helps improve the performance of PITF.

## 2.2 Sequential Recommendation

Another relevant study is the next item prediction from sequential user interaction logs [19]. The goal is to recommend items that match a given sequence of user actions. The work [9] is the first to employ the recurrent neural network to tackle session-based recommendation. Since the attention mechanism can automatically assign different influences (weights) of items to capture the dynamic property, recently, some papers exploit it in sequence-aware recommendation. The work [25] used it for next-item recommendation in the transactional context, where the attention mechanism is used to calculate the relevant items. In [2], the user memory network with attention mechanism is used to explain that how user’s historical records affect current decisions. The work [30] introduces an attention-based user behavior modeling framework and [28] proposes a novel two-layer hierarchical attention network, which is used to couple user long-term and short-term preferences. However, these works are all item recommender systems and no work uses the attention mechanism in tag recommendation.

## 3 Notations and Preliminaries

In this section, we first formalize the problem of the tag recommendation, and then shortly recapitulate the pairwise interaction tensor factorization since this paper also uses the pairwise interaction model.

### 3.1 Problem Statements

Let  $\mathcal{U}$ ,  $\mathcal{I}$  and  $\mathcal{T}$  denote the set of all users, all items and all tags respectively. Given the user  $u$  and the item  $i$ , the task of tag recommendation is to find a list  $\mathcal{T}(u, i) \subseteq \mathcal{T}$  that the user  $u$  want to annotate the item  $i$ . Usually, it is formulated as a ranking problem which is to predict the order of the user’s tagging preference in the  $\mathcal{T}$  [20, 24]. So we could define the list of the Top-N tags as:

$$Top(u, i, N) := \arg \max_{t \in \mathcal{T}}^N \hat{y}_{u,i,t},$$

where  $\hat{y}_{u,i,t}$  denotes the prediction score of triple  $\langle u, i, t \rangle$ .

Moving one step forward, since tags the user used are often related to previous behaviors and the timestamp, we use  $L$  to denote the list the user used and  $s$  to denote the timestamp. Then the score we need to predict is

$$Top(u, i, s, l_u^s, N) := \arg \max_{t \in \mathcal{T}}^N \hat{y}_{u,i,t}^{s, l_u^s},$$

where  $l_u^s \subseteq L$  denotes the tag list that the user used before the timestamp  $s$ .

### 3.2 Pairwise Interaction Tensor Factorization (PITF)

Given the user-item pair  $(u, i)$ , the PITF model predicts a scoring function  $\hat{Y} : \mathcal{U} \times \mathcal{I} \times \mathcal{T} \rightarrow R$  which can be used to derive an order that trivially satisfies antisymmetry and transitivity [21], where  $\hat{Y}$  is a three dimensional tensor. Then to get the  $\hat{y}_{u,i,t}$ , this model supposes the interaction between the pair  $u$  and  $i$  has no impact on the final result, and has the equation as

$$\hat{y}_{u,i,t} = \sum_k \hat{u}_{u,k} \cdot \hat{t}_{t,k}^U + \sum_k \hat{i}_{i,k} \cdot \hat{t}_{i,k}^I$$

where  $\hat{u}_{u,k}$  and  $\hat{i}_{i,k}$  are latent vector elements of the user  $u$  and the item  $i$  respectively.  $\hat{t}_{t,k}^U$  and  $\hat{t}_{i,k}^I$  are both latent vector elements of the tag  $t$ , but they are used in different relations of user-tag and item-tag.  $k$  is the dimensionality of latent vectors.

PITF is a special case of the Canonical Decomposition (CD) model with dimensionality  $2 \cdot k$  [21]. It explicitly models the two-way interactions of user-tag and item-tag. In this paper, we will further explore the representation of users, items and tags.

## 4 Attention-Based Neural Tag Recommendation

In this section, we first present our general framework and compare it with the PITF model. Then we propose an attention-based model to learn the user-tag representation. Lastly, a multi-layer perceptron (MLP) is introduced to capture the pairwise interactions among users, items and tags.

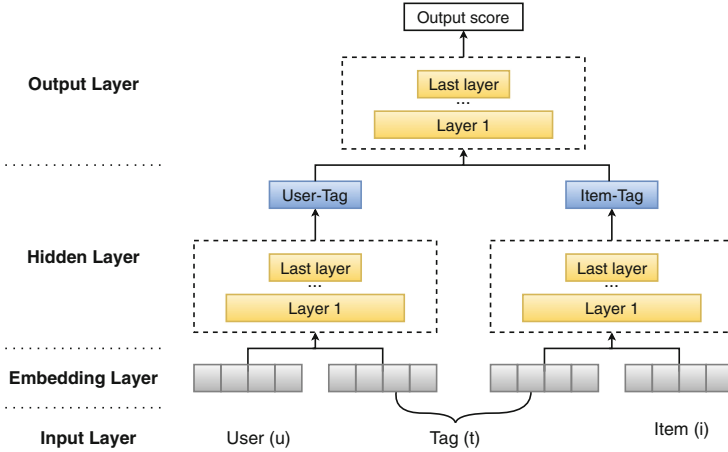
### 4.1 General Framework

As illustrated in Fig. 1, the general framework of the proposed model has four layers and the target of modeling is to estimate the score of alternative tags. This framework does not take the dynamic at different time steps into account.

The bottom input layer consists of three binarized sparse vectors with one-hot encoding that denote user  $u$ , item  $i$  and tag  $t$ , respectively. Similar to the one-hot encoding of words in natural language processing, there is an embedding layer above the input layer, which is a fully connected layer to embed these one-hot representations into the continuous low-dimensional space. Note that the tag will have two different dense vectors since the user-tag interaction is different from the item-tag interaction. Formally, the embedding layer consisting of matrices is:

$$U \in \mathbb{R}^{|\mathcal{U}| \times k}, I \in \mathbb{R}^{|\mathcal{I}| \times k}, T^U \in \mathbb{R}^{|\mathcal{T}| \times k}, T^I \in \mathbb{R}^{|\mathcal{T}| \times k}. \quad (1)$$

Then the user embedding  $v_u$  and the tag embedding  $v_t^u$  are fed into a hidden layer to get a vector that represents the user and tag interaction. The hidden layer is a customized multi-layer network which is used to capture features of



**Fig. 1.** The architecture of the general framework. It consists of four layers

user-tag interactions. The item embedding  $v_i$  and the tag embedding  $v_t^i$  will be processed in the same way. Formally,

$$v_{u,t} = \phi_{last}^{u,t} (\dots \phi_2^{u,t} (\phi_1^{u,t} (v_u, v_t^u)) \dots) \tag{2}$$

$$v_{i,t} = \phi_{last}^{i,t} (\dots \phi_2^{i,t} (\phi_1^{i,t} (v_i, v_t^i)) \dots) \tag{3}$$

where  $v_{u,t}$  and  $v_{i,t}$  are vectors representing user-tag interactions and item-tag interaction respectively.  $\phi_x$  denotes the mapping function for the  $x$ -th network layer. Note that the dimension of vectors of users, items and tags may be different from  $v_{u,t}$  and  $v_{i,t}$ .

The final layer of output is also a multi-layer network,

$$y_{u,i,t} = \phi_{last}^{u,i,t} (\dots \phi_2^{u,i,t} (\phi_1^{u,i,t} (v_{u,t}, v_{i,t})) \dots), \tag{4}$$

since  $y_{u,i,t}$  denotes the score of the tag  $t$ , this network should map tow vectors into a numerical value. What needs a special attention is that  $\phi_x$  could be the neural network or any other form, e.g., element-wise product.

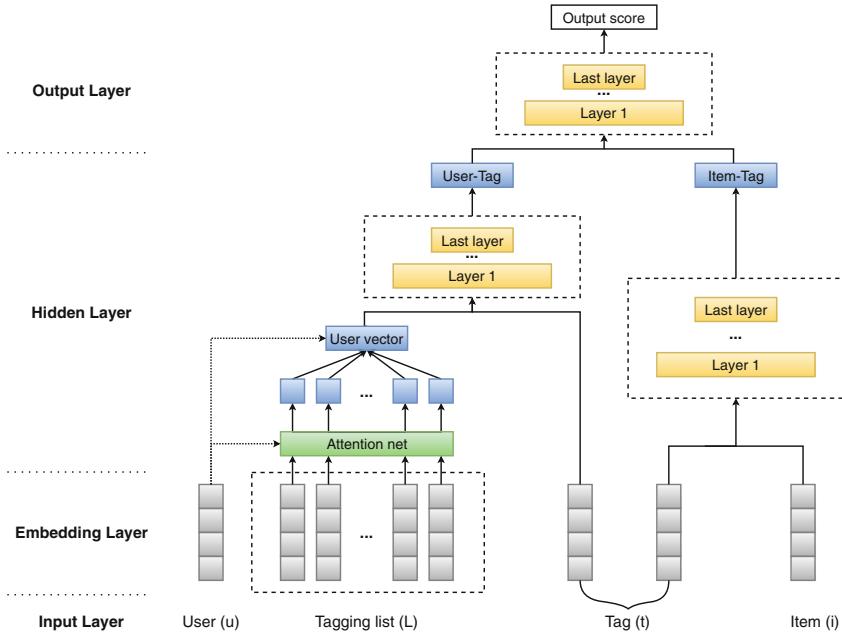
Some studies have demonstrated that the traditional matrix factorization is equivalent to a three-layer network [7, 8], which has the input layer and embedding layer to embed users and items into dense low-rank vectors and uses the inner product in the output layer. Similarly, the PITF can be also interpreted as a special case of the general framework. For the hidden layer, the PITF is defined as

$$v_{u,t} = \phi_1^{u,t} (v_u, v_t^u) = v_u \odot v_t^u$$

$$v_{i,t} = \phi_1^{i,t} (v_i, v_t^i) = v_i \odot v_t^i$$

where  $\odot$  denotes the element-wise product of vectors. Then the output layer is

$$y_{u,i,t} = \phi_1^{u,i,t} (v_{u,t}, v_{i,t}) = v_{u,t} + v_{i,t}.$$



**Fig. 2.** The architecture of ABNT, which has a attention network to learn the user representation

Under this framework, the PITF model can be easily extended. For example, we could re-weight the contribution of user-tag and item-tag interactions as

$$y_{u,i,t} = \phi_1^{u,i,t}(v_{u,t}, v_{i,t}) = w_{u,t} \cdot v_{u,t} + w_{i,t} \cdot v_{i,t} \tag{5}$$

where  $w_{u,t}$  is the weight of user-tag and  $w_{i,t}$  is the weight of item-tag. It can be learned from the data or handcrafted.

### 4.2 ABNT: Attention-Based Neural Tag Recommendation Model

In this section, we introduce a novel approach based on the general framework and the attention mechanism according to the following characteristics of the user’s tagging behavior. Firstly, the user’s tagging behavior is dynamic at different time steps, and then for a user, different tags have different influences on his choices. Finally, for different users, same tags may have different impacts on the next tag choice by them.

As shown in Fig. 2, compared with the general framework proposed above, the ABNT model has a user tag list in the input layer and an attention-based pooling layer in the hidden layer. More specially, the tags in the list will also be embedded into dense vectors. Since we only focus on the user’s tagging behavior, these tags will use the same mapping matrix with the  $v_t^u$ . Then the attention net is employed to infer the weight guided by the user embedding and the weight is

used to get the user representation to endow it more information about tagging sequence. Next, each part of ABNT will be introduced in details.

**Input and Embedding Layer.** Different from the general framework, the tagging list  $l_u^s$  before the time stamp  $s$  is given in the input layer. Supposing the length of  $l_u^s$  is  $m$ , the input of this model is  $(u, i, t, s, (t_1^u, t_2^u, \dots, t_m^u))$ , where  $t$  is the tag of which the score we want to estimate.

The embedding layer is the same as shown in Expression 1. Tags in  $(t_1^u, t_2^u, \dots, t_m^u)$  are also embedded into vectors by  $T^u$ .

**Hidden Layer.** In the hidden layer, the attention mechanism is applied for computing the importance of each tag in the  $l_u^s$ , and then aggregates the embedding of these tags to form the user preference representation. The attention network can be formulated as

$$\begin{aligned} h_j^u &= a(W_{att} \cdot v_j^u + b_{att}) \\ \alpha_j &= \frac{\exp(u^\top h_j)}{\sum_{p \in l_u^s} \exp(u^\top h_p)} \end{aligned} \quad (6)$$

where the first formula is a fully connected layer.  $W_{att} \in \mathbb{R}^{k \times k}$  and  $b_{att} \in \mathbb{R}^{k \times 1}$  are the model parameters.  $v_j^u$  is the embedding vector of  $t_j^u$  in the  $l_u^s$ , which is fed into the connected layer to get a hidden representation  $h_j^u$ .  $a$  is the activation function and we utilize ReLU to enhance nonlinear capability in this paper. To capture the information of the user's preference, the user embedding vector  $u$  is used to be the context vector to measure the attention score with  $h_j$ . The softmax function is employed to calculate attention weight  $\alpha_j$ . Then the tagging behavior representation  $v_b^u$  is

$$v_b^u = \sum_{j \in l_u^s} \alpha_j \cdot v_j^u \quad (7)$$

It is the sum of tags embedding in the tagging list weighted by the attention score, and we defined the user representation as follows

$$v_u^{hybrid} = a(W_{hybrid} \cdot \begin{bmatrix} v_u \\ v_b^u \end{bmatrix}) + b_{hybrid} \quad (8)$$

where the  $W_{hybrid} \in \mathbb{R}^{2k \times k}$  and  $b_{hybrid} \in \mathbb{R}^{k \times 1}$  are model parameters. The two vectors  $v_u$  and  $v_b^u$  will be concatenated to learn a better representation. We note that some of sequence models such as recurrent neural network can be used there and we leave the exploration as a future work.

Then in this model, the vector denoting the interaction between  $u$  and  $t$  is defined as

$$v_{u,t} = a_{last}(W_{last}^{u,t}(\dots a_2(W_2^{u,t}(W_1^{u,t} \cdot \begin{bmatrix} v_u^{hybrid} \\ v_t^u \end{bmatrix}) + b_1^{u,t}) + b_2^{u,t}) \dots) + b_{last}^{u,t} \quad (9)$$

where  $W_x^{u,t}$ ,  $b_x^{u,t}$  and  $a_x^{u,t}$  denote the weight matrix, bias vector and activation function, respectively. Since we only focus on the user's tagging behavior, the way to learn  $v_{i,t}$  is the same as Eq. 3, and the hidden layer is a standard MLP like the Eq. 9. It is formulated as

$$v_{i,t} = a_{last}(W_{last}^{i,t}(\dots a_2(W_2^{i,t}(W_1^{i,t} \cdot \begin{bmatrix} v_i \\ v_t^i \end{bmatrix} + b_1^{i,t}) + b_2^{i,t})\dots) + b_{last}^{i,t}) \quad (10)$$

The last layer will map latent vectors of the user-tag and the item-tag interaction to a numerical value in the proposed model. In other words, supposing the dimension of latent vectors in the last layer but one is  $f$ , parameters of the last layer are

$$W_{last} \in \mathbb{R}^{f \times 1}, b_{last} \in \mathbb{R}^{1 \times 1} \quad (11)$$

**Output Layer.** Following the previous work [24], considering the both time-awareness and personalization aspects, the output layer of this proposed model is defined as

$$\hat{y}_{u,i,t}^{s,l_u^s} = w_{u,t}^s \cdot v_{u,t} + w_{i,t}^s \cdot v_{i,t} \quad (12)$$

where  $v_{u,t}$  is calculated by the Eq. 9 and  $v_{i,t}$  is calculated by the Eq. 10.  $w_{u,t}$  and  $w_{i,t}$  are weights of the user-tag embedding and item-tag embedding, respectively. More specifically, it is an extended re-weight model of PITF.

Intuitively, the higher frequency of the past occurrences of the tag, the larger weight it should have. So this model uses the same way as time-aware PITF model introduced in [24] to calculate weights in advance. Formally, weights are defined as

$$w_{u,t}^s = 1 + \log_{10}(1 + 10^\alpha \cdot \|\tau(u, t, s)\|) \quad (13)$$

$$w_{i,t}^s = 1 + \log_{10}(1 + 10^\alpha \cdot \|\|\hat{Y}_{i,t}\|\|) \quad (14)$$

where constant  $\alpha$  is used to control the growth rate of the weight.  $|\hat{Y}_{i,t}|$  is the frequency that the item  $i$  is annotated by the tag  $t$  and  $\|\|\hat{Y}_{i,t}\|\|$  is its normalized value. For the weight of user-tag interaction,  $\|\tau(u, t, s)\|$  is the normalized intensity value of the event at time  $s$ , and the formulation of it is given as follows,

$$\tau(u, t, s) = \tau_0 + \sum_{s_i < s} \exp^{-d(s-s_i)} \quad (15)$$

where  $d$  is the intensity parameter and  $s_i$  is the time stamp of the tagging event before  $s$ . More information about this can be found in [24].

### 4.3 Model Inference

The task of this work is to provide a ranked list of tags given  $(u, i, l_u^s, s)$ . Following the previous work [21, 24], the pair-wise ranking objective function is used to learn parameters of this model. It utilizes the BPR optimization criterion [20] and assumes that users prefer the next tag to be used instead of other unobserved

**Table 1.** Properties of two datasets,  $|\mathcal{U}|$  is the number of users,  $|\mathcal{I}|$  is the number of resources,  $|\mathcal{T}|$  is the number of tags.

Datasets	$ \mathcal{U} $	$ \mathcal{I} $	$ \mathcal{T} $	#train samples	#test samples
Movielens	984	5852	8778	45034	1507
LastFM	1803	12504	9701	180396	5947

tags. Supposing  $t_{neg}$  is the negative tag which is sampled from unobserved tags, the objective is defined as

$$\arg \min_{\Theta} \sum_{(u,i,t,t_{neg},s,l_u^s) \in D} -\ln \sigma(\hat{y}_{u,i,t}^{s,l_u^s} - \hat{y}_{u,i,t_{neg}}^{s,l_u^s}) + \lambda \|\Theta\| \quad (16)$$

where  $\Theta$  is the set of parameters we need to estimate,  $D$  is the set of training samples,  $\sigma$  is the logistic function, and  $\lambda$  is the regularization parameter. all activation functions of the neural network in this paper is ReLU, which is more biologically plausible and proven to be non-saturated [5]. Since the derivative of parameters can be calculated with standard back-propagation,  $\Theta$  is updated with gradient descent.

## 5 Experiments

In this section, we evaluate the proposed model on two real-world datasets. We first introduce the setup of experiments and then report the experimental results. We further discuss the influence of the components in the proposed model and the length of the tagging list.

### 5.1 Experimental Setup

**Datasets.** We evaluate the proposed method on two real-world datasets [14], Movielens and LastFM<sup>1</sup>. The domain of Movielens is the movie and of LastFM is music. Since the proposed model uses the tagging list to learn the user’s embedding vector, we filtered out users whose number of tagging less than 3. The relevant statistical properties are described in Table 1 after filtering.

Since different users have the different number of tagging behaviors and too long-term records may have minimal impact on the present, we set the length of tagging list a constant for each record in the experiment. Specially, supposing that  $l_u = (t_1, t_2, t_3, \dots, t_{|l_u|})$  is the tagging list for the user  $u$  and  $m$  denote the constant of the length of tagging list. Without loss of generality, for a record  $(u, i, t_j, s)$ , we generate the sample as  $(u, i, t_j, s, l_u^s)$ , where  $l_u^s = (t_{j-m-1}, t_{j-m}, \dots, t_{j-1})$  and  $j$  denotes the  $j$ -th tagging. If  $j < m$ , we pad with zero and ignore it when training the model.

<sup>1</sup> Both datasets can be found in <http://files.grouplens.org/datasets/hetrec2011>.



**Table 2.** Some of hyper-parameters of PITF, TAPITF and the proposed model ABNT.  $k$  is the latent factor dimension, in other words, it is the parameter of embedding layer in Eq. 1.  $\lambda$  is the regularization parameter,  $d$  is defined in Eq. 15.

Models \ Parameters	$k$	$\lambda$	learning rate	# negative samples	$d$
PITF	64	0.00005	0.05	100	-
TAPITF	64	0.00005	[0.001,0.0005,0.0001]	10	0.5
ABNT	64	0.00005	[0.001,0.0005,0.0001]	10	0.5

Then we use the same protocol as described in [24], which split train and test sets by leave-one-out evaluation. For each user, we put tags of the last item the user annotated into the test set and utilize the remaining data for training. If the user only annotated one item, his tagging records are all put into the test set so that he is a cold-start user.

**Metrics.** To evaluate the performance of the proposed method, we adopt two widely used metrics:

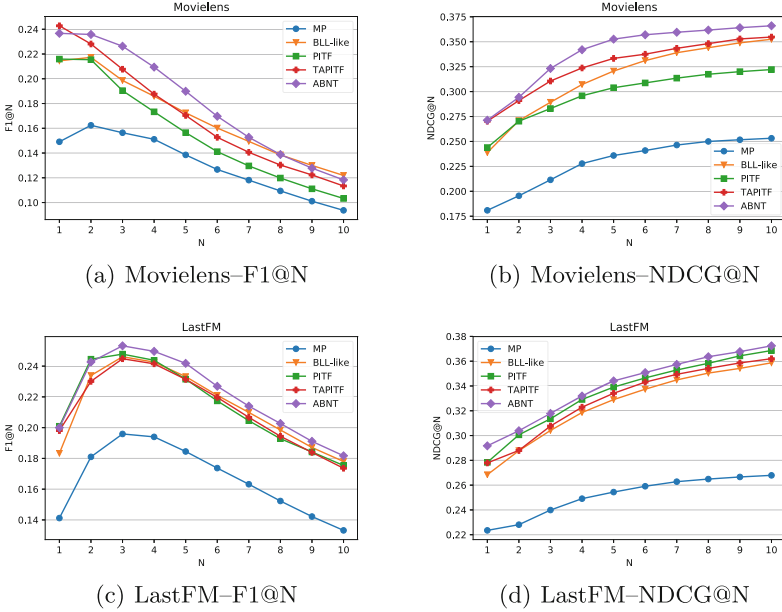
$$F1@N := \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}@N$$

$$NDCG@N := \frac{DCG}{IDCG}@N$$

The F1 score is the harmonic average of the precision and recall of the test sample, and the NDCG, which is Normalized Discounted Cumulative Gain, accounts for the position of the result by assigning higher scores to the correct ranking [6]. The larger metric values indicate better performances and we report the average score for all samples in the test set.

**Baseline.** We compared the proposed methods ABNT with the following methods:

- MP. Tags are ranked by the popularity judged by the number of use.
- PITF [21]. This method is pairwise interaction tensor factorization model which modifies traditional tensor factorization approach and has been introduced in Sect. 3.2.
- BLL-like. We use  $BLL_{AC} + MP_m$  [13], which estimates the probability of a tag being applied by a particular user as a function of usage frequency and recency of the tag in the user’s past behavior.
- TAPITF [24]. This method extends PITF by adding weights to user-tag interaction and item-tag interaction respectively and exploits Hawkes process with the exponential intensity to calculate the use-tag weight.



**Fig. 3.** Performance of Top-N tag recommendation where K ranges from 1 to 10 on the two datasets

**Parameter Setting.** We implemented our proposed methods based on Pytorch. For  $BLL_{AC} + MP_m$  methods, we follow the setting in [14]. As we discussed above, PITF and TAPITF are both special cases of our proposed general framework, following the setting in [24], some of the hyper-parameters are described in Table 2. For TAPITF and ABNT, we initialize their embedding parameters using the pre-trained models, and we tune the learning rate of  $[0.001, 0.0005, 0.0001]$ . We also tune the length of  $l_u^s$  of  $[5, 10, 15, 20, 25, 30]$  for ABNT. Another important hyper-parameter is  $\alpha$ , which is the growth rate in Eqs. 13 and 14. We test it of  $[0.5, 1, 1.2, 1.4, 1.6, 1.8, 2]$  and report the best value for the model TAPITF and ABNT. In order to reduce the complexity, we employ three MLP layers for learning intermediate representation vectors like the work [8], and one full connected layer for get the final representation values in Eqs. 2 and 3. In other words, the dimension of hidden vectors is  $128 \rightarrow 64 \rightarrow 32 \rightarrow 1$ .

## 5.2 Comparison of Performance

Figure 3 shows the performance of Top-N recommended list where N ranges from 1 to 10. Different from item recommendation which may offer users many items on the web page, tag recommender systems often do not give users too many options. So the N we set is not too big.

As can be seen, ABNT outperforms all other methods in most cases. It demonstrates the superiority of the proposed model. Especially, on MovieLens

**Table 3.** Influence of components at F1@5 and NDCG@5. Non-Neural is the model which we only used the attention mechanism and Non-Attention is only MLP used in.

Dataset	Method	F1@5	NDCG@5
MovieLens	TAPITF	0.1704	0.3333
	Non-Neural	0.1724	0.3264
	Non-Attention	0.1828	0.3451
	ABNT	<b>0.1899</b>	<b>0.3526</b>
LastFM	TAPITF	0.2314	0.3341
	Non-Neural	0.2366	0.3381
	Non-Attention	0.2411	0.3403
	ABNT	<b>0.2418</b>	<b>0.3441</b>

dataset, ABNT improves 10.1% and 5.9% in terms of F1@5 and NDCG@5 compared with the second best methods, i.e., TAPITF or *BLL*-like. Note that in reality the recommendation system offers not only one option, we think the result has more practical significance when  $N = 5$ . For baseline methods, TAPITF outperforms *BLL*-like methods at NDCG but not always better at F1. When  $N$  is larger than 5, the value of TAPITF is worse than *BLL*-like at F1. Compared with PITF, the relative performance improvement by TAPITF is 8.88% at F1@5 and 9.67% at NDCG@5.

On LastFM dataset, all methods have similar results except MP, but the proposed model has the relative improvement over the best baseline 3.69% at F1@5 and 1.47% at NDCG@5. Surprisingly, PITF model surpasses TAPITF in most case on this dataset, and we attribute it to not enough temporal information by only re-weighting the contribution of user-tag and item-tag interactions.

### 5.3 Influence of Components

The contribution of each component for the final result is shown in Table 3. Especially, On both datasets, the impact of neural networks on results is pronounced compared with TAPITF, which improves 7.23%, 4.19% at F1@5 and 3.54%, 2.01% at NDCG@5 on MovieLens and LastFM, respectively. This indicates that the hidden layer consisting of MLP layers could learn a better representation of interactions among users, items and tags. Also, it shows the interactions may have complex non-linearities.

Then the attention network could also improve the result of the model. For example, Non-Neural achieves better performance at F1@5 on both MovieLens and LastFM datasets compared with TAPITF. Then after using the attention network, ABNT outperforms Non-Attention on two metrics. Values of F1@5 and NDCG@5 are 0.1899 and 0.3526 on MovieLens dataset and 0.2418, 0.3431 on LastFM dataset. The reason may be that the attention network has further captured the dynamic performances and complex patterns in the sequence of users.

**Table 4.** Influence of the length.

Metrics	5	10	15	20	25	30
MovieLens						
F1@5	0.1862	<b>0.1899</b>	0.1864	0.1857	0.1869	0.1865
NDCG@5	0.3503	<b>0.3525</b>	0.3519	0.3403	0.3506	0.3407
LastFM						
F1@5	<b>0.2418</b>	0.2388	0.2374	0.2394	0.2376	0.2411
NDCG@5	<b>0.3441</b>	0.3391	0.3294	0.3369	0.3284	0.3375

#### 5.4 Influence of the Length of Tagging List

The length of the tagging list is the unique hyper-parameter in our proposed model ABNT. The influence of it has been summarized in Table 4. As shown in this table, when the length is 10 for MovieLens and 5 for LastFM, the model have the best performance. It is not the longer length the better performance in our model. Intuitively, this phenomenon can be explained that users usually have a strong trend to reused the recently used tags, so the shorter list may have enough information for recommendation.

On the contrary, the longer list may have more useless information and reduce the performance of the model. We think that this is a deficiency of the attention mechanism. Essentially, the attention mechanism still assigns a small weight to the irrelevant record in the sequence, which adds the noise for the training of the model. When the length of the sequence increases, the irrelevant record may also increase, causing the useless information to accumulate. This observation has also verified the rationality of BLL-like methods at a certain extent since they are based on users' recent tagging behaviors.

## 6 Conclusion

In this paper, we explored neural networks and the attention mechanism for the tag recommendation. We proposed a general framework and an attention-based method – ABNT, which captures the complex pattern of the user's tagging sequence to represent the user embedding vectors and model the non-linearity of interactions among users, items and tags. From the experiment, we observed the better performance of the proposed model compared with all baselines.

In the future work, we will study the sequence modeling for the user embedding vector such as recurrent neural networks and extend ABNT to model more information of interactions between items and tags.

**Acknowledgement.** We thank the reviewers for their valuable and helpful comments. This work was supported by National Key R&D Program of China (No. 2017YFC0803700), NSFC grants (No. 61532021), Shanghai Knowledge Service Platform Project (No. ZF1213) and SHEITC.

## References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate (2014). arXiv preprint: [arXiv:1409.0473](https://arxiv.org/abs/1409.0473)
2. Chen, X., et al.: Sequential recommendation with user memory networks. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, pp. 108–116. ACM (2018)
3. Du, N., Wang, Y., He, N., Sun, J., Song, L.: Time-sensitive recommendation from recurrent user activities. In: Advances in Neural Information Processing Systems, pp. 3492–3500 (2015)
4. Fang, X., Pan, R., Cao, G., He, X., Dai, W.: Personalized tag recommendation through nonlinear tensor factorization using Gaussian kernel. In: AAAI, pp. 439–445 (2015)
5. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pp. 315–323 (2011)
6. He, X., Chen, T., Kan, M.Y., Chen, X.: TriRank: review-aware explainable recommendation by modeling aspects. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, pp. 1661–1670. ACM (2015)
7. He, X., Du, X., Wang, X., Tian, F., Tang, J., Chua, T.S.: Outer product-based neural collaborative filtering (2018). arXiv preprint: [arXiv:1808.03912](https://arxiv.org/abs/1808.03912)
8. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web, pp. 173–182. International World Wide Web Conferences Steering Committee (2017)
9. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based recommendations with recurrent neural networks (2015). arXiv preprint: [arXiv:1511.06939](https://arxiv.org/abs/1511.06939)
10. Hotho, A., Jäschke, R., Schmitz, C., Stumme, G.: Information retrieval in folksonomies: search and ranking. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 411–426. Springer, Heidelberg (2006). [https://doi.org/10.1007/11762256\\_31](https://doi.org/10.1007/11762256_31)
11. Jäschke, R., Marinho, L., Hotho, A., Schmidt-Thieme, L., Stumme, G.: Tag recommendations in folksonomies. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenić, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 506–514. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-74976-9\\_52](https://doi.org/10.1007/978-3-540-74976-9_52)
12. Koren, Y.: Collaborative filtering with temporal dynamics. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 447–456. ACM (2009)
13. Kowald, D., Kopeinik, S., Seitlinger, P., Ley, T., Albert, D., Trattner, C.: Refining frequency-based tag reuse predictions by means of time and semantic context. In: Atzmueller, M., Chin, A., Scholz, C., Trattner, C. (eds.) MSM/MUSE 2013. LNCS (LNAI), vol. 8940, pp. 55–74. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-14723-9\\_4](https://doi.org/10.1007/978-3-319-14723-9_4)
14. Kowald, D., Lex, E.: Evaluating tag recommender algorithms in real-world folksonomies: a comparative study. In: Proceedings of the 9th ACM Conference on Recommender Systems, pp. 265–268. ACM (2015)

15. Kowald, D., Seitlinger, P., Kopeinik, S., Ley, T., Trattner, C.: Forgetting the words but remembering the meaning: modeling forgetting in a verbal and semantic tag recommender. In: Atzmueller, M., Chin, A., Scholz, C., Trattner, C. (eds.) MSM/MUSE 2013. LNCS (LNAI), vol. 8940, pp. 75–95. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-14723-9\\_5](https://doi.org/10.1007/978-3-319-14723-9_5)
16. Kowald, D., Seitlinger, P., Trattner, C., Ley, T.: Long time no see: the probability of reusing tags as a function of frequency and recency. In: Proceedings of the 23rd International Conference on World Wide Web, pp. 463–468. ACM (2014)
17. Lin, Z., et al.: A structured self-attentive sentence embedding (2017). arXiv preprint: [arXiv:1703.03130](https://arxiv.org/abs/1703.03130)
18. Marinho, L.B., Schmidt-Thieme, L.: Collaborative tag recommendations. In: Preisach, C., Burkhardt, H., Schmidt-Thieme, L., Decker, R. (eds.) Data Analysis, Machine Learning and Applications. Studies in Classification, Data Analysis, and Knowledge Organization, pp. 533–540. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-78246-9\\_63](https://doi.org/10.1007/978-3-540-78246-9_63)
19. Quadrana, M., Cremonesi, P., Jannach, D.: Sequence-aware recommender systems (2018). arXiv preprint: [arXiv:1802.08452](https://arxiv.org/abs/1802.08452)
20. Rendle, S., Balby Marinho, L., Nanopoulos, A., Schmidt-Thieme, L.: Learning optimal ranking with tensor factorization for tag recommendation. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 727–736. ACM (2009)
21. Rendle, S., Schmidt-Thieme, L.: Pairwise interaction tensor factorization for personalized tag recommendation. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining, pp. 81–90. ACM (2010)
22. Seitlinger, P., Kowald, D., Trattner, C., Ley, T.: Recommending tags with a model of human categorization. In: Proceedings of the 22nd ACM International Conference on Information and Knowledge Management, pp. 2381–2386. ACM (2013)
23. Symeonidis, P., Nanopoulos, A., Manolopoulos, Y.: Tag recommendations based on tensor dimensionality reduction. In: Proceedings of the 2008 ACM Conference on Recommender Systems, pp. 43–50. ACM (2008)
24. Wang, K., Jin, Y., Wang, H., Peng, H., Wang, X.: Personalized time-aware tag recommendation. In: AAAI Conference on Artificial Intelligence (2018)
25. Wang, S., Hu, L., Cao, L., Huang, X., Lian, D., Liu, W.: Attention-based transactional context embedding for next-item recommendation. AAAI (2018)
26. Wang, W., Zhang, W., Wang, J., Yan, J., Zha, H.: Learning sequential correlation for user generated textual content popularity prediction. In: IJCAI, pp. 1625–1631 (2018)
27. Wu, C.Y., Ahmed, A., Beutel, A., Smola, A.J., Jing, H.: Recurrent recommender networks. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, pp. 495–503. ACM (2017)
28. Ying, H., et al.: Sequential recommender system based on hierarchical attention networks. In: 27th International Joint Conference on Artificial Intelligence (2018)
29. Zhang, L., Tang, J., Zhang, M.: Integrating temporal usage pattern into personalized tag prediction. In: Sheng, Q.Z., Wang, G., Jensen, C.S., Xu, G. (eds.) APWeb 2012. LNCS, vol. 7235, pp. 354–365. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-29253-8\\_30](https://doi.org/10.1007/978-3-642-29253-8_30)
30. Zhou, C., et al.: ATRank: an attention-based user behavior modeling framework for recommendation (2017). arXiv preprint: [arXiv:1711.06632](https://arxiv.org/abs/1711.06632)



# Density Matrix Based Preference Evolution Networks for E-Commerce Recommendation

Panpan Wang<sup>1</sup>, Zhao Li<sup>2</sup>(✉), Xuming Pan<sup>2</sup>, Donghui Ding<sup>2</sup>, Xia Chen<sup>2</sup>,  
and Yuexian Hou<sup>1</sup>(✉)

<sup>1</sup> Tianjin University, Tianjin, China  
{panpan.tju,yxhou}@tju.edu.cn

<sup>2</sup> Alibaba Group, Hangzhou, China  
{lizhao.lz,xuming.panxm,donghui.ddh,xia.cx}@alibaba-inc.com

**Abstract.** In e-commerce platforms, mining temporal characteristics in user behavior is conducive to recommend the right product for the user at the right time. Recently, recurrent neural networks (RNNs) based methods have achieved profitable performance in exploring temporal features, however, in complex e-commerce scenarios, user preferences changing over time have not been fully exploited. In order to fill the gap, we propose a novel representation for user preferences with the inspiration of a quantum concept, **density matrix**. It encodes a mixture of item subspaces and represents distribution of user preferences at one time stamp. Further, such a representation and RNNs are combined to form our proposed Density Matrix based Preference Evolution Networks (**DMPENs**). Experiments on Amazon datasets as well as real-world e-commerce datasets demonstrate the effectiveness of the proposed methods, which achieve rapid convergence and superior performance compared with the state-of-the-art methods in terms of AUC and accuracy.

**Keywords:** E-commerce recommendation ·  
Recurrent neural networks · Density matrix

## 1 Introduction

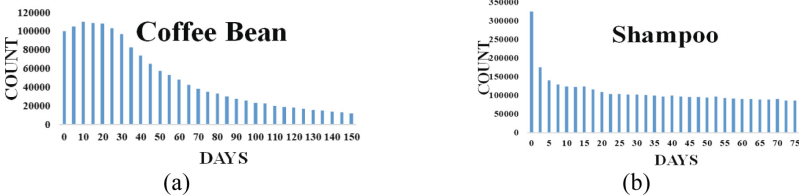
Temporal dynamics is a typical feature of purchase behavior in the e-commerce platform. For example, as the time changing from summer to winter, users' shopping characteristics shift from cool to warm. In order to capture temporal characteristics, recurrent neural networks (RNNs) based approaches have been recently proposed and demonstrated their competitive capacity to recommend proper items for users at the right time [19, 26]. For example, RNNs exhibit better performance than DNN, SVD, and time-SVD++ in sequence prediction tasks and session-based recommendation tasks [6, 23, 25]. Subsequently, some machine learning or statistical learning methods, such as attention mechanism [26], matrix factorization [19], and point process [4], were assisted with

purchasing sequences	probability
EA→FU→DM	8%
FU→EA→DM	11%
EA→DM→FU	14%
FU→DM→EA	16%
DM→EA→FU	25%
DM→FU→EA	26%

**Fig. 1.** Probability statistics about six purchasing sequences derived from three products, i.e. electric appliance (EA), furniture (FU) and decoration material (DM)

RNNs to further promote recommendation. Although these RNNs-based methods achieved profitable performance, temporal characteristics of user preference have not been fully exploited yet, especially in complex e-commerce scenarios.

User behavior is highly complex and diverse in real e-commerce scenarios. To illustrate this, we survey the purchase order among electric appliance (EA), furniture (FU) and decoration material (DM) on Taobao website, which is the largest e-commerce platform in China. The statistical results are shown in the Fig. 1. The average time interval between any two categories of them is 2-3 months. In general, we believe that most of users would buy DM to renovate the house, and then purchase EA and FU. However, the statistic results show that only about 51% of users choose to buy DM first. 30% of users choose to buy DM after purchasing EA or FU, and about 19% users consider to buy DM at last. Intuitively, the diversity of purchase order is not explicitly observed.



**Fig. 2.** Statistics about repurchase cycles of shampoo and coffee bean from Taobao website.

In addition, the repurchase is also an important feature that the e-commerce recommendation system needs to capture. It can avoid recommending similar products that users have just bought. However, this feature is also diverse in complex e-commerce scenarios. To this end, we also collect data from Taobao and make statistics about the distribution of repurchase cycle of coffee bean and shampoo, as shown in the Fig. 2. It can be seen that the repurchase cycle distribution of shampoo is significantly smoother than that of coffee bean. It illustrates that users choose to buy shampoo again at any time. In contrast, most users repurchase coffee bean within 100 days. The diversity of repurchase



may limit currently proposed RNNs based methods to effectively capture user preferences.

In this paper, we aim to address the above problem with the support of quantum theory, which has been successfully applied in the field of information science [1, 14, 21]. Density matrix, a concept describing the quantum state statistically, plays a key role in currently proposed quantum theory inspired methods. One of the representative tasks is the quantum language model proposed for traditional information retrieval task [14]. In this method, density matrix was used to represent user information needs and the evaluated documents, and then the relative entropy was used to calculate the similarity as the document ranking score. The density matrix is also in conjunction with Convolutional Neural Network (CNN) to mine the correlation between questions and answers. However, to the best of our knowledge, existing works fail to theoretically explain the effectiveness of density matrix based representation for learning embedding and training neural network.

From the above analysis, both density matrix and RNNs have their advantages, in which the former has strong representation ability, the latter effectively captures temporal features. We thus combine the advantages of both density matrix and RNNs to model the evolution of user preferences in complex e-commerce scenarios, and propose **Density Matrix based Preference Evolution Networks (DMPENs)**. Specifically, our proposed methods make the following contributions:

- **A novel user preference representation based on density matrix.** In previous RNNs based methods, features were first mapped into low-dimensional embedding vectors and then fed to RNNs. In this paper, we convert these vectors into density matrices before feeding them to RNNs. Compared with the embedding vector, the density matrix takes into account the 2-order correlation between original embedding entries. The correlation makes the update of one embedding entry affect other entries, and is able to improve the preference representation of RNNs.
- **Integrating density matrix into RNNs.** In this article, we integrate the density matrix directly into three kinds of RNNs, i.e. basic-RNN [18], GRU [3], and LSTM [7]. The BPTT algorithm [17] for training RNNs is also applicable. In order to preserve 2-order correlations and avoid losing information, we do not choose to lower dimension of density matrix. Observed from the experimental results, we find that the convergence speed of DMPENs is over three times faster than that of other compared methods, which indicates density matrix is very effective and efficient for all RNN variables.
- **Effective methods with a rapid convergence and a significant improvement.** We evaluate our proposed DMPENs on Amazon product datasets and Taobao e-commerce datasets of high complexity and variety over time. A series of systematic experiments have shown that our proposed methods achieve a rapid convergence and also significantly outperform the state-of-the-art methods in terms of **Area Under the Curve (AUC)** and **Accuracy**. The high precision of our proposed methods on predicting user

preferences illustrates that density matrix effectively captures the temporal diversity of user behavior, e.g., repurchasing cycle in e-commerce platforms described above.

We clarify that our motivation of using a concept in quantum theory is to inspire new perspective and formulation for the application of user preferences prediction, instead of developing quantum computation algorithms. We also remind that density matrix based representation is totally different from a feature combination method proposed by Product-based Neural Networks (PNN) [13], in which outer product is conducted between different features' embeddings. By contrast, outer product utilized in our proposed methods is conducted on one feature embedding itself. To the best of our knowledge, this is the first attempt to integrate density matrix into RNNs for recommendation task.

## 2 Related Work

In this section, we introduce the application of density matrix in several information science fields, as well as methods for predicting the evolution of user preferences based on RNNs.

**Density matrix based representation** was first rising in Information Retrieval (IR). van Rijsbergen argued that density matrix can be thought of a generalized query [15]. Then, queries were exactly expressed as density matrices [11]. Later, density matrix was further utilized to model term dependencies with the property of quantum entanglement [20]. For more complex search scenarios, e.g. session search, density matrix was used to model the transformation of user information needs [9, 12]. Except for IR, it was extended into natural language processing (NLP), in which the density matrix was employed to encode dependency neighborhoods and represent sentence sequence [1, 2]. In the work of sentiment analysis, density matrix was also developed to model correlations between images and texts [22]. Although the density matrix based representation was well applied in the field of information science, it has not been introduced into recommendation task. Its application in this paper makes a preliminary attempt. **RNNs based methods** have been successively applied to model temporal dependencies on sequential user behavior. Except for the previous described work, a variant of LSTM was proposed to model the time interval between users' actions and improved the recommendation performance [25]. Later, an adapted Attention-GRU model with three important modifications was proposed for brand-level ranking system in e-commerce platform [26]. Besides, RNNs were utilized in a session-based recommendation task that user records were created in a session-align way and put into multi-layer GRU architectures [6]. These three state-of-the-art methods will be compared with our proposed methods in the experiment section. In this paper, we will combine the advantages of both density matrix and RNNs to improve the accuracy of user preferences prediction in complex e-commerce scenarios.

### 3 Density Matrix Based Preference Evolution Networks

In this section, we describe our model in the context of user preferences prediction, which aims at improving prediction accuracy based on sequential user behavior. Specifically, we introduce our model in the following steps. First, we introduce preliminary of density matrix. Next, we design a density matrix based representation for user preferences. Then, we show how to integrate such a representation into RNNs architectures. The detail is described below.

#### 3.1 Preliminary of Quantum Theory

The mathematical formalism of quantum theory is based on linear algebra. Now, we briefly introduce some basic concepts that will be utilized in this paper. In quantum theory, quantum state refers to the state of a quantum system. It provides a probability distribution for the outcome of each possible measurement on the system. A basic quantum state can be described by a unit vector,  $\mathbf{u} \in \mathbb{R}^{n \times 1}$  over an Hilbert space  $\mathbb{H}$ . In the common mathematical formalism of quantum theory,  $\mathbf{u}$  is often denoted by  $|u\rangle$  with the Dirac's notation, called a *ket*. Its transpose  $\mathbf{u}^\top$  is denoted by  $\langle u|$ , called a *bra*. The projector  $\Pi$  onto the direction  $|u\rangle$  is represented by  $|u\rangle\langle u|$ , i.e.  $\Pi = |u\rangle\langle u|$ , which is an outer product (also called *dyad*) of  $|u\rangle$  itself. If a quantum state can be represented as a vector, it is always called a pure state. However, a quantum system can be in a statistical ensemble of different state vectors. For example, there may be a 50% probability that the state vector is  $|u_1\rangle$  and a 50% chance that the state vector is  $|u_2\rangle$ . This means that the system is in a mixed state. The density matrix is especially useful for describing the mixed state. A density matrix  $\rho$  is defined as a mixture of  $|u_i\rangle\langle u_i|$ , i.e.  $\rho = \sum_i p_i \Pi_i = \sum_i p_i |u_i\rangle\langle u_i|$ , where  $p_i$  represents the probability of  $|u_i\rangle\langle u_i|$ . In general, both pure and mixed state can be characterized by a single density matrix. When the density matrix is applied into data science, it is necessary to first determine the object it describes. For instance, in quantum language model [14], the object described is text, which is a mixture of query terms. In e-commerce scenarios, user's selection of products typically reflect their preferences. In this paper, user preferences are the objects we aim to describe, and they can be regarded as a mix of selected products.

#### 3.2 Density Matrix Based Representation for User Preferences

User preferences at one time stamp are characterized by the mix of selected items. Following this idea, we first implement the vector representation of each selected item. Item features are commonly mapped into lower embedding vectors in currently developed deep learning based methods [24]. In this paper, item features are also transformed in the same way. We construct a items' embedding matrix, denoted as  $E \in \mathbb{R}^{|V| \times d}$ , where  $|V|$  is the length of items and  $d$  is the dimension of embedding. Note that, the whole embedding matrix is initialized randomly, but can be updated during the training process. Each item embedding is denoted as  $\vec{e}_i^\top = (e_{i1}, e_{i2}, \dots, e_{id}) \in E$ . Since each item is served as a

pure state, which is an unit state vector, we should first normalize each item embedding vector ( $\vec{e}_i^T \in E$ ) as follow:

$$|e_i\rangle = \frac{\vec{e}_i}{\|\vec{e}_i\|_2}, \quad (1)$$

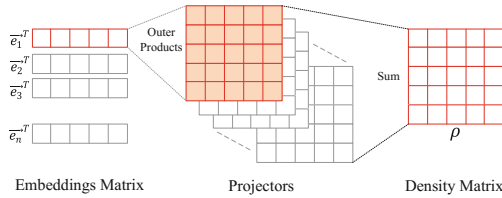
where  $|e_i\rangle = (e'_{i1}, e'_{i2}, \dots, e'_{id})^T$ . Then the corresponding projector  $\Pi_i$ , i.e. a subspace spanned by the embedding-based state vector  $|e_i\rangle$ , is computed by an outer product as follow:

$$\Pi_i = |e_i\rangle\langle e_i| = \begin{pmatrix} e'_{i1} \\ e'_{i2} \\ \dots \\ e'_{id} \end{pmatrix} \times (e'_{i1}, e'_{i2}, \dots, e'_{id}) = \begin{bmatrix} (e'_{i1})^2 & e'_{i1}e'_{i2} & \dots & e'_{i1}e'_{id} \\ e'_{i2}e'_{i1} & (e'_{i2})^2 & \dots & e'_{i2}e'_{id} \\ \vdots & \dots & \dots & \vdots \\ e'_{id}e'_{i1} & e'_{id}e'_{i2} & \dots & (e'_{id})^2 \end{bmatrix} \quad (2)$$

From this equation, it is easily observed that the projector contains **2-order correlation**, for example,  $e'_{i1}e'_{i2}$ , between any two entries of  $|e_i\rangle$ . Next, user preference corresponding to a mixed state at one time stamp is represented by a density matrix, which is derived as:

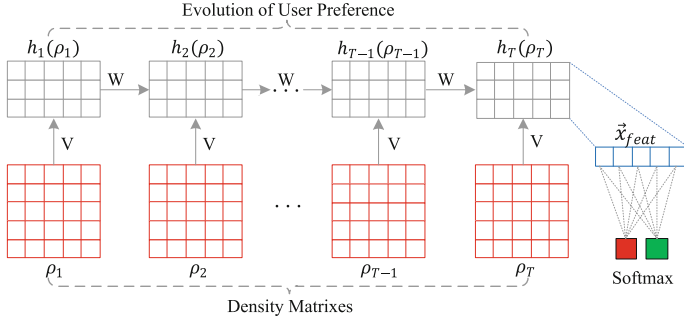
$$\rho = \sum_i p_i \Pi_i = \sum_i p_i |e_i\rangle\langle e_i| = \begin{bmatrix} \sum_i p_i (e'_{i1})^2 & \sum_i p_i e'_{i1}e'_{i2} & \dots & \sum_i p_i e'_{i1}e'_{id} \\ \sum_i p_i e'_{i2}e'_{i1} & \sum_i p_i (e'_{i2})^2 & \dots & \sum_i p_i e'_{i2}e'_{id} \\ \vdots & \dots & \dots & \vdots \\ \sum_i p_i e'_{id}e'_{i1} & \sum_i p_i e'_{id}e'_{i2} & \dots & \sum_i p_i (e'_{id})^2 \end{bmatrix} \quad (3)$$

where  $p_i$  stands for the selected frequency of item  $i$ , and  $\sum_i p_i = 1$ . It is clearly that density matrix is symmetric. This process for constituting a density matrix is also illustrated in Fig. 3.



**Fig. 3.** Density matrix based representation

Distinct from currently proposed RNNs-based methods, which directly align the embedding vector for each item, the mixture of the subspaces spanned by the embedding vectors has not been taken into consideration. From Eqs. (2) and (3), it can be observed that each entry in the normalized embedding is multiplied with other entries under the formalization of density matrix. In other



**Fig. 4.** Density Matrix based Preference Evolution Network

words, the density matrix contains more correlation features than the embedding vector. Among RNNs based methods, the embedding vector is trainable, in which each entry in it is considered as a variable and is updated as the RNN is trained. Each embedding entry is independent during the training process such that it is updated without affecting other entries. However, by constructing a density matrix, each entry is correlated with other entries, causing the update of one entry affecting other entries that are multiplied with it. This correlation may be beneficial for learning the embedding matrix, and even training RNNs architectures. Here, we make a hypothesis that density matrix based representation can assist RNNs to accelerate the convergence of the training process. This hypothesis will be confirmed in the experimental part.

### 3.3 Learning Evolution of User Preferences

In this paper, recurrent neural networks (RNNs) are employed to model the evolution of user preferences. Specifically, we use density matrix  $\rho_t \in \mathbb{R}^{d \times d}$  obtained by Eq. (3) to represent user preference at time  $t$ .  $\rho_t$  is then fed into RNNs. Although the input is a matrix, the training algorithm (back-propagation through time, BPTT) remains available. For simplicity, we take a basic-RNN [18] as an example. The updating form of hidden state  $h_t(\rho_t) \in \mathbb{R}^{k \times d}$  is described below:

$$h_t(\rho_t) = \sigma(\mathbf{W}h_{t-1}(\rho_{t-1}) + \mathbf{V}\rho_t + b) \tag{4}$$

where  $\mathbf{W} \in \mathbb{R}^{k \times k}$ ,  $\mathbf{V} \in \mathbb{R}^{k \times d}$  are weight parameters, and  $b \in \mathbb{R}^{k \times d}$  denotes the bias.  $\sigma(\cdot)$  is an activation function, such as commonly used Tanh or Sigmoid function (Tanh is utilized in this paper). Equation (4) can be replaced with updating functions adopted in other kinds of RNNs, for example, LSTM [7] and GRU [3].

For the predicted item, it is first formed with a density matrix and then as input into the last time stamp  $T$  of RNNs. The corresponding hidden state  $h_T(\rho_T)$  is regarded as feature for prediction. Due to the matrix form of  $h_T(\rho_T)$ , for convenient, we accumulate it into a vector, denoted as  $\mathbf{x}_{feat} \in \mathbb{R}^{1 \times d}$  simply

by rows. And then putting it into a fully-connected layer, as shown in Fig. 4. The softmax activation is adopted, and outputs the probabilities of both positive label (i.e. being selected) and negative label (i.e. not being selected) of the predicted item. If necessary, the probability of the positive sample can be served as ranking score in real-world application scenarios, for example, recommending top-k items for the user. The back propagation is trained with the cross entropy loss:

$$\mathbf{L} = - \sum_i^N [y_i \log p(\mathbf{x}_{feat}) + (1 - y_i) \log(1 - p(\mathbf{x}_{feat}))] \quad (5)$$

where  $p(\mathbf{x}_{feat})$  is probability output by softmax, and  $y_i$  represents the target label. Our proposed density matrix based representation can be applied to any kinds of RNNs architectures to model user preferences, these integrated methods are collectively named as **Density Matrix based Preference Evolution Networks (DMPENs)**. Figure 4 summarizes the basic architecture of our proposed DMPENs. We also abstract the above approach in Algorithm 1.

---

**Algorithm 1.** DMPEN: Density Matrix based Preference Evolution Network

---

**Input:**

$|V|$ : length of items;  $d$ : dimension of embedding.

**Output:**

$p(\mathbf{x}_{feat})$ : predicted probability for feature  $\mathbf{x}_{feat}$ .

- 1: Initialize an embedding matrix  $E \in \mathbb{R}^{|V| \times d}$ ;
  - 2: **for** each time stamp  $t \in [1, T]$  **do**
  - 3:   Obtain user’s selected item set  $C_t$  at time  $t$ ;
  - 4:   **for** each item  $i \in C_t$  **do**
  - 5:     Normalize the embedding  $|e_i\rangle$  using Eq.(1);
  - 6:     Calculate the frequency  $p_i$  of item  $i$ ;
  - 7:     Construct projector  $\Pi_i$  by Eq.(2);
  - 8:   **end for**
  - 9:   Derive density matrix  $\rho_t$  via Eq.(3);
  - 10:   Update hidden state in Eq.(4);
  - 11: **end for**
  - 12: Acquire the last hidden state  $h_T(\rho_T)$  as a feature vector  $\mathbf{x}_{feat}$ ;
  - 13: Put  $\mathbf{x}_{feat}$  into a fully-connected layer, and the softmax activation is adopted to compute  $p(\mathbf{x}_{feat})$ .
- 

### 3.4 A Time-Align Way for Capturing Repurchase Characteristic

To facilitate the training of RNNs, the sequence length (i.e. the number of hidden state) is often fixed. However, the length of input sequence is usually unequal to RNNs’ length. In order to solve the mismatch between lengths, a method of directly padding zeroes into input is commonly adopted. This does not fit our task that our goal is capturing the temporal characteristic, such as repurchase

cycle, in complex e-commerce scenarios. Let’s take an example to illustrate this. One purchase sequence of user A is {‘i1’, ‘i2’, ‘i3’, ‘i1’}. User B has another purchase sequence of {‘i1’, ‘i4’, ‘i5’, ‘i1’}. Intuitively, both A and B have the same repurchase interval of ‘i1’, because there are two items before repurchasing ‘i1’ in both sequences. Actually, user A may repurchase ‘i1’ after 2 weeks, while user B purchases ‘i1’ again after 2 months. The two repurchase periods are quite different. Padding zeroes directly would make no sense in this paper. Therefore, we adopt a time-align way to obtain input sequence. User behavior data at each day is recorded in the log of e-commerce website. We first sort logs in chronological order. Then, we extract each user record from logs and sort them by time. If the day has no behavior, it will be filled with zero. After preprocessing, all user sequences are of equal length. The above process is also described in Fig. 5.  $u_{i,j}$  indicates the record of  $i^{th}$  user on the  $j^{th}$  day.

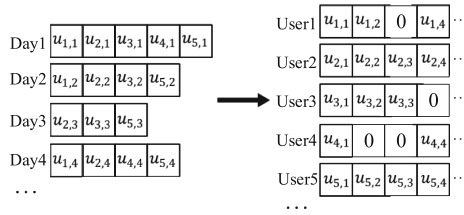


Fig. 5. A time-align user sequence creation

## 4 Experiments

In this section, we present the experiments in detail, including datasets, preprocessing, experimental setups, comparison models, and results analysis. Experiments on a public dataset as well as a real-world dataset collected from Taobao, one of the biggest e-commerce platform in China, demonstrate the effectiveness of DMPENs which outperform the state-of-the-art methods on user preferences prediction task.

### 4.1 Datasets, Preprocessing and Statistics

**Amazon Dataset**<sup>1</sup>. It contains product reviews and metadata from Amazon, which is used as benchmark dataset [5, 10, 16, 24]. We conduct experiments on a subset named ‘Clothing, Shoes and Jewelry’. In general, the user can make a comment on the product after purchasing it. It means that the dataset only contains the purchase records, which can directly reflect repurchase characteristics in user behavior. More statistics about Amazon dataset are summarized in Table 1.

<sup>1</sup> <http://jmcauley.ucsd.edu/data/amazon/>.

**Table 1.** Characteristics of three datasets. (Taobao datasets are collected in 2018).

Dataset	Amazon-I	Amazon-II	Taobao-I	Taobao-II	Taobao-III
Users	10044	6792	164244	161051	216726
Categories	538	499	10707	11502	12153
Train set	7/10/1999-23/07/2014		13/06-13/07	14/04-13/06	13/02-14/05
Test set			14/07-13/08	14/06-13/08	15/5-13/08
Train size	135016	88854	645844	628484	914070
Test size	174	506	209188	196392	245946
PreInHis.train	2.63%	5.60%	9.10%	13.07%	15.20%
PreInHis.test	0.5%	10.98%	12.35%	16.04%	17.85%

**Taobao Dataset<sup>2</sup>.** It is collected from Taobao, the largest e-commerce website in China. We select 974,702 users from user pool and extract user behavior logs from February 8, 2018 to August 13, 2018. In this dataset, we will consider scenarios that are more complex than Amazon dataset. Repurchase is not only explicitly reflected by purchase behavior, but also implicitly related with clicks, collections and add-to-cart. The e-commerce website is used for online shopping as well as providing a wealth of product information for user to reference. Users may browse the website to obtain product information, and then choose other online websites or offline shops to purchase. Therefore, it is necessary to consider such an implicit repurchase scenario. To this end, we extract user’s clicking, collecting, purchasing, and adding-to-cart, these four kinds of records, to capture both explicit and implicit repurchase characteristics. More statistics are summarized in Table 1.

**Preprocessing.** Product information, such as ‘product\_ids’, is particularly sparse for crawling repurchase feature. The user buys some products with different product\_ids, but in reality these products belong to the same category. In order to capture the repurchase characteristics of user behavior, we only extract the ‘category\_id’ as a feature to predict user preferences. The time-align method described above is used to preprocess user sequence consisting of ‘category\_id’. To confirm that our model can grasp the long-term preference with different time lengths, we further divide each user sequence into subsequences according to the time length of 60-days, 120-days and 180-days, which correspondingly form three sub-datasets, denoted by I, II, and III respectively. For each sub-dataset, the first half of each subsequence is served as train sample, and the second half is used as test sample. After that, the time lengths of both train set and test set under three sub-datasets are 30 days, 60 days, and 90 days, respectively. Each day is a time stamp. Categories at the last time stamp in both train set and test set are used as positive samples. The negative samples are equal with positive samples, and randomly selected from category pool by removing positive samples. We

<sup>2</sup> <https://www.taobao.com/>.



filter out users who have no record at the last time stamp in both train and test set. Since Amazon dataset generates too few test sample with the 180-day duration (only one sample), we decided to remove this sub-dataset and reserve ‘Amazon-I’ and ‘Amazon-II’ with 60 days and 120 days, respectively. For the Taobao dataset, we separately take the last 60 days, the last 120 days and the last 180 days from all 187 days to form ‘Taobao-I’, ‘Taobao-II’ and ‘Taobao-III’.

**Statistics.** We summarize the statistics of five datasets in detail, as shown in Table 1. PreInHis.train and PreInHis.test indicate the average proportion of predicted categories appeared in the historical categories on train set and test set, respectively. The values of both Amazon and Taobao are less than 20%, indicating the diversity of user preferences. Volume of Taobao Dataset is much larger than Amazon, which brings more challenges.

## 4.2 Compared Methods, Implementation and Evaluation Metrics

In this paper, we integrate density matrix based representation into three kinds of RNNs, i.e. basic-RNN, GRU, and LSTM. According to the combined RNN architectures, we divide all of compared methods into the following groups. The details are described as:

**Basic-RNN, DMPEN-RNN:** Basic-RNN is the popular architecture for modeling temporal dynamics of long-term user preferences [4, 23]. DMEPN-RNN combines the density matrix based representation with basic-RNN.

**GRU, Session-RNN, Attention-GRU-3M, DMPEN-GRU:** GRU is commonly used popular architecture for modeling long-term user preferences. Session-RNN exploits a three-layer **GRU** structure to capture user’s short-term interest based on sequential actions within a session [6]. In this paper, we consider the log recorded in one day on the e-commerce platform as a session. A mini-batch is aligned with user\_id. That is, a mini-batch is actually a user record. Attention-GRU-3M is designed for a brand-level ranking system in e-commerce platform, and has achieved three modifications based on Attention-GRU model [26]. The time interval is set to 1. The dimension of attention cell is set to 10. The dimension of dense layer is set to 128. It would be a competitive method in this paper. DMPEN-GRU integrates density matrix based representation with GRU.

**LSTM, Time-LSTM, DMPEN-LSTM:** LSTM is the representative RNNs architecture. Time-LSTM equips LSTM with time gates to model real time interval feature [25]. Since real time interval is not taken into consideration in our proposed methods, for a fair comparison, time interval of each user record is set to 1. DMPEN-LSTM integrates density matrix based representation with LSTM.

All methods are implemented based on TensorFlow platform. We set the hidden state dimension of all utilized RNNs to 100 and the embedding dimension to 20. We use ADAM [8] as the optimizer with an initial learning rate of 0.001.

For Amazon datasets, we set the mini-batch size to 50 and the epoch to 10. For Taobao datasets, we set the mini-batch size to 500 and the epoch to 50.

To test whether these methods can correctly predict user preference (i.e. categories) at the last time stamp, Area Under ROC Curve (**AUC**) and Accuracy (**ACC**) are employed as the evaluation metrics.

### 4.3 Results Analysis on Amazon Datasets

In this section, we analyze the experimental results and the convergence among all methods on Amazon-I and Amazon-II. Table 2 displays the evaluation results across Amazon datasets in terms of AUC and ACC. It can be seen that DMPENs almost achieve the best results among all groups. Specifically, for basic-RNN based group, DMPEN-RNN has large improvements against baseline method over two datasets under the evaluation of two metrics. For Amazon-I, DMPEN-RNN is increased by 5.98% under AUC and 5.75% under ACC. For Amazon-II, DMPEN-RNN is significantly improved by 9.67% under AUC. These three results have the greatest improvements in comparison of other groups. In addition, DMPEN-RNN achieves the best performance of 0.9800 against all compared methods on Amazon-I in terms of ACC.

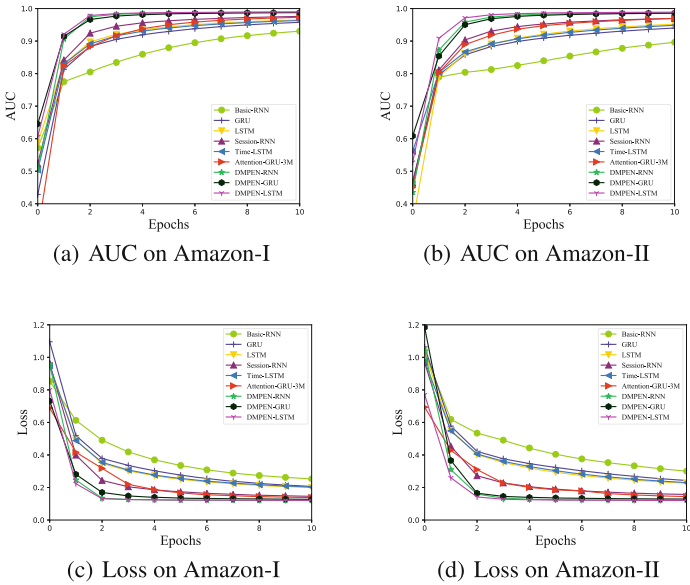
**Table 2.** Model comparison on Amazon datasets. (Bold typeset indicates the best performance among compared groups. (·) denotes the improved percentage of the DMPEN based method against the Baseline in the corresponding group. Session-RNN is implement with multi-layer GRU cell.)

Methods	Amazon-I		Amazon-II	
	AUC	ACC	AUC	ACC
Basic-RNN (Baseline)	0.9332	0.9267	0.9004	0.9100
DMPEN-RNN	<b>0.9890</b>	<b>0.9800</b>	<b>0.9875</b>	<b>0.9300</b>
	(5.98%)	(5.75%)	(9.67%)	(2.20%)
GRU (Baseline)	0.9593	0.9667	0.9422	0.9180
Session-RNN	0.9765	0.9400	0.9706	0.9320
Attention-GRU-3M	0.9742	0.9800	0.9708	0.9360
DMPEN-GRU	<b>0.9876</b>	0.9600	<b>0.9859</b>	<b>0.9620</b>
	(2.95%)	(-0.96%)	(4.64%)	(4.79%)
LSTM (Baseline)	0.9664	0.9533	0.9536	0.9400
Time-LSTM	0.9646	0.9400	0.9503	0.9180
DMPEN-LSTM	<b>0.9895</b>	<b>0.9667</b>	<b>0.9890</b>	0.9340
	(2.39%)	(1.40%)	(3.71%)	(-0.64%)

For GRU-based methods, DMPEN-GRU is slightly lower than GRU on Amazon-I under AUC, but it is further improved compared to GRU under the rest of metrics and datasets. And DMPEN-GRU achieves the best result

of 0.9620 under ACC on Amazon-II. Session-RNN has improvements over GRU, mainly because of using a three-layer GRU structure. Attention-GRU-3M achieves better results than Session-RNN, indicating that the attention mechanism can assist to further enhance the performance of GRU. Although it achieves the best performance with ACC on Amazon-I, but it is still lower than DMEPN-GRU under the rest of evaluation metrics and datasets.

For LSTM-based group, DMPEN-LSTM obtains improvements by more than 1.40% against LSTM. It also achieves the best performance of 0.9895 and 0.9890 among all groups across two datasets in terms of AUC. Time-LSTM always performs worse than LSTM, indicating that setting the real time interval to 1 makes no effect.



**Fig. 6.** The performance comparisons over epochs of all methods with respect to AUC and cross entropy loss on Amazon-I and Amazon-II datasets. DMPENs achieve higher AUC and lower cross entropy loss than their competitors.

The upper of Fig. 6 describes the AUC curves of all the compared methods on Amazon-I and Amazon-II. It can be easily observed that DMPEN based methods exhibit faster convergence than other compared methods. They can get a stable convergence within 4 epochs, however, other methods even can not get a convergence after 10 epochs, such as, basic-RNN, GRU, and LSTM. DMPEN-LSTM has the most rapid convergence across both datasets. All kinds of DMPENs start to coincide as epoch is larger than 3. Session-RNN converges faster than Attention-GRU-3M, but both of them almost coincide with each other at last. LSTM performs better than GRU and basic-RNN, and has a similar trend of convergence with Time-LSTM, due to the useless of its time gate.

The bottom two subfigures describe the cross entropy loss curves performed on Amazon-I and Amazon-II. DMPENs also achieve the rapidest convergence, which is similar to AUC curves. They get a stable convergence after running 4 epochs. DMPEN-LSTM obtains the lowest loss after 10 epochs. Attention-GRU-3M has a slower speed than Session-RNN at the beginning on both datasets, but it reaches a lower loss at last. Basic-RNN still performs the worst. LSTM converges faster than basic-RNN and GRU. Time-LSTM remains the same trends with LSTM on both datasets.

#### 4.4 Results Analysis on Taobao Datasets

The experimental results of Taobao datasets are shown in Table 3. For basic-RNN based group, DMPEN-RNN achieves the maximum increment over 17.0% against basic-RNN compared with the improvements in other groups across three datasets in terms of AUC and ACC. It can be attributed to the weak memory of basic-RNN.

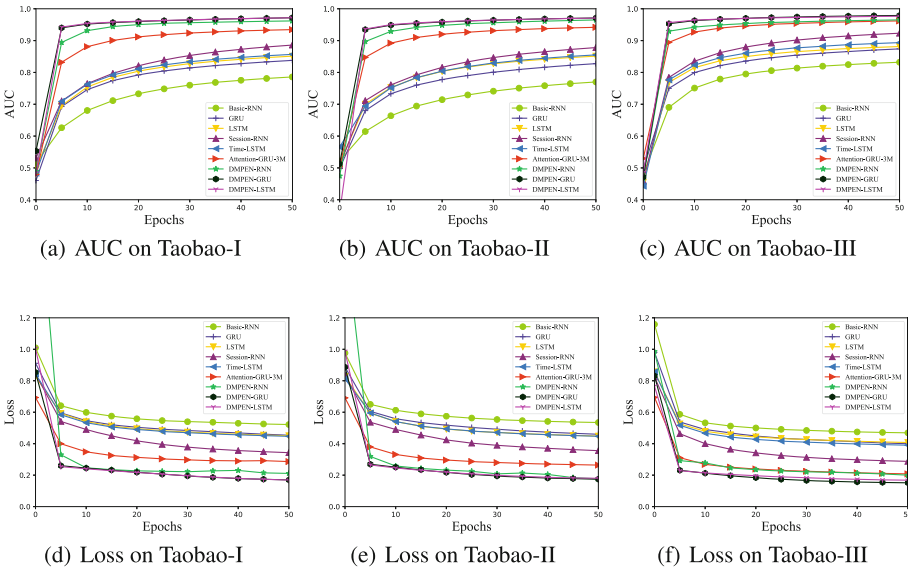
**Table 3.** Model comparison on Taobao datasets. (Bold typeset indicates the best performance among their compared groups. (·) denotes the improved percentage of the DMPEN based method against the Baseline in the corresponding group. Session-RNN is implement with multi-layer GRU cell.)

Methods	Taobao-I		Taobao-II		Taobao-III	
	AUC	ACC	AUC	ACC	AUC	ACC
Basic-RNN (Baseline)	0.7288	0.7371	0.7706	0.7260	0.8323	0.7330
DMPEN-RNN	<b>0.9584</b>	<b>0.8680</b>	<b>0.9557</b>	<b>0.8520</b>	<b>0.9662</b>	<b>0.8587</b>
	(31.50%)	(17.76%)	(24.2%)	(17.36%)	(18.09%)	(17.15%)
GRU (Baseline)	0.8383	0.7773	0.8282	0.7668	0.8744	0.7890
Session-RNN	0.8865	0.8582	0.8787	0.8525	0.9236	0.8580
Attention-GRU-3M	0.9347	0.8674	0.9420	0.8496	0.9621	0.8539
DMPEN-GRU	<b>0.9601</b>	0.8640	<b>0.9594</b>	<b>0.8660</b>	<b>0.9687</b>	<b>0.8612</b>
	(14.53%)	(11.15%)	(15.84%)	(12.94%)	(10.78%)	(10.04%)
LSTM (Baseline)	0.8517	0.7919	0.8516	0.7823	0.8817	0.7893
Time-LSTM	0.8572	0.7938	0.8556	0.7987	0.8940	0.8058
DMPEN-LSTM	<b>0.9692</b>	<b>0.8821</b>	<b>0.9697</b>	<b>0.8782</b>	<b>0.9774</b>	<b>0.8645</b>
	(13.80%)	(11.39%)	(13.87%)	(12.26%)	(10.85%)	(9.53%)

Among GRU based methods, DMPEN-GRU achieves the largest improvement over other methods, although it is slightly lower than Attention-GRU-3M in terms of ACC on Taobao-I. To be specific, compared with GRU, DMPEN-GRU achieves improvement over 10% across three datasets with respect to AUC and ACC. All evaluation results of Session-RNN is better than GRU. Attention-GRU-3M exceeds GRU and Session-RNN with respect to AUC.

For all of LSTM based methods, DMPEN-LSTM has a remarkable improvement of 13.8% (AUC) and 11.39% (ACC) on Taobao-I, 13.87% (AUC) and 12.26% (ACC) on Taobao-II, and 10.85% (AUC) and 9.53% (ACC) on Taobao-III against LSTM. DMPEN-LSTM achieves the best performance over all the other compared methods, but has a relatively lower improvement than DMPEN-RNN. This is because LSTM is more effective than basic-RNN to memorize long-term user preferences. Time-LSTM has a similar performance with LSTM, which is consistent with the performance of Amazon datasets.

From both Amazon datasets and Taobao datasets, DMPENs almost achieve large improvements over other baseline methods and state-of-the-art methods. It verifies the effectiveness of our proposed density matrix based representation, which the 2-order correlation between two embedding entries provides useful patterns for predicting user preferences changing over time.



**Fig. 7.** The performance comparisons over epochs of all methods with respect to AUC and cross entropy loss on Taobao-I, Taobao-II, and Taobao-III datasets. DMPENs achieve higher AUC and lower cross entropy loss than their competitors.

As analyzed above, density matrix based representation can significantly improve the prediction performance. Next, we further investigate the advantage of density matrix, and compare the convergence performance in terms of AUC and loss, as illustrated in Fig. 7. For AUC curves, DMPENs reach a stable convergence after running 10 epochs across three datasets. Attention-GRU-3M can reach a stable convergence when epoch reaches 40. For other compared methods, they still have a gentle upward trend after 50 epochs. Although their performance

could be further improved with much more epochs, the convergence is too slow to reach. LSTM, Time-LSTM and GRU have similar convergence trends due to their analogical architectures.

For loss curves, DMPENs achieve the rapidest convergence after 10 epochs. Especially for DMPEN-GRU and DMPEN-LSTM can get stable convergence when epoch is around 5 among three datasets. DMPEN-RNN has lower loss than other non-DMPEN methods across Taobao-I and Taobao-II. Attention-GRU-3M exhibits a similar loss curve with DMPEN-RNN on Taobao-III. Session-RNN apparently performs better than Time-LSTM, and other RNNs. Time-LSTM has a slightly lower loss than LSTM. From the figures, LSTM, Time-LSTM, GRU and Session-RNN have not reached a stable convergence at the end of training. It shows the complexity of user behavior in Taobao datasets, but verifies the robustness of our proposed methods. The good performance of DMPENs mainly benefits from the representation of density matrix, in which each entry is correlated with other entries in the same embedding, and its update has an effect on other entries. In contrast, each entry in embedding based representation is independent, in which the update of one entry does not affect other entries. Therefore, the convergence of DMPENs is largely accelerated.

## 5 Conclusions

In this paper, we propose Density Matrix based Preference Evolution Networks (DMPENs) which integrate the density matrix, a key concept in quantum theory, into RNNs architectures and combine with a time-align data-preprocess method to model complex temporal characteristics in the evolution of user preferences. Our proposed density matrix based representation encodes user preferences at one time stamp into subspaces, in which 2-order correlation between embedding entries is involved. Since the update of each entry in density matrices can affect other entries, it further accelerates the convergence of RNNs. DMPENs are evaluated on Amazon datasets and real-world e-commerce datasets. The loss curves show that DMPENs obtain a stable convergence after running 3 or 5 epochs across all datasets, however, other compared methods even can not achieve a convergence after 10 epochs or 50 epochs. Under the evaluation of AUC and ACC, DMPENs realize a significant improvement by approximately from 2% to 32% against all baseline methods. In particular, DMPEN with LSTM almost exhibits the best performance compared with the state-of-the-art methods across all datasets.

**Acknowledgement.** This work is funded in part of the National Key R&D Program of China (2017YEF0111900), the National Natural Science Foundation of China (61876129), the National Natural Science Foundation of China (Key Program, U1636203), the Alibaba Innovation Research Foundation 2017 and the European Unions Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 721321. Part of the work was performed when Panpan Wang visited the Alibaba Inc. in 2018.

## References

1. Basile, I., Tamburini, F.: Towards quantum language models. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 1840–1849 (2017)
2. Blacoe, W., Kashefi, E., Lapata, M.: A quantum-theoretic approach to distributional semantics. In: Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 847–857 (2013)
3. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling (2014). arXiv preprint: [arXiv:1412.3555](https://arxiv.org/abs/1412.3555)
4. Dai, H., Wang, Y., Trivedi, R., Song, L.: Deep coevolutionary network: embedding user and item features for recommendation (2016). arXiv preprint: [arXiv:1609.03675](https://arxiv.org/abs/1609.03675)
5. He, R., McAuley, J.: Ups and downs: modeling the visual evolution of fashion trends with one-class collaborative filtering. In: Proceedings of the 25th International Conference on World Wide Web, pp. 507–517. International World Wide Web Conferences Steering Committee (2016)
6. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based recommendations with recurrent neural networks (2016)
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.* **9**(8), 1735–1780 (1997)
8. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization (2014). arXiv preprint: [arXiv:1412.6980](https://arxiv.org/abs/1412.6980)
9. Li, Q., Li, J., Zhang, P., Song, D.: Modeling multi-query retrieval tasks using density matrix transformation. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 871–874. ACM (2015)
10. McAuley, J., Targett, C., Shi, Q., Van Den Hengel, A.: Image-based recommendations on styles and substitutes. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 43–52. ACM (2015)
11. Piwowarski, B., Frommholz, I., Lalmas, M., Van Rijsbergen, K.: What can quantum theory bring to information retrieval. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, pp. 59–68. ACM (2010)
12. Piwowarski, B., Lalmas, M.: A quantum-based model for interactive information retrieval. In: Azzopardi, L., et al. (eds.) ICTIR 2009. LNCS, vol. 5766, pp. 224–231. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-04417-5\\_20](https://doi.org/10.1007/978-3-642-04417-5_20)
13. Qu, Y., et al.: Product-based neural networks for user response prediction over multi-field categorical data (2018). arXiv preprint: [arXiv:1807.00311](https://arxiv.org/abs/1807.00311)
14. Sordoni, A., Nie, J.Y., Bengio, Y.: Modeling term dependencies with quantum language models for IR. In: Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 653–662. ACM (2013)
15. Van Rijsbergen, C.J.: The Geometry of Information Retrieval. Cambridge University Press, Cambridge (2004)
16. Veit, A., Kovacs, B., Bell, S., McAuley, J., Bala, K., Belongie, S.: Learning visual clothing style with heterogeneous dyadic co-occurrences. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 4642–4650 (2015)

17. Werbos, P.J.: Backpropagation through time: what it does and how to do it. *Proc. IEEE* **78**(10), 1550–1560 (1990)
18. Williams, R.J., Zipser, D.: A learning algorithm for continually running fully recurrent neural networks. *Neural Comput.* **1**(2), 270–280 (1989)
19. Wu, C.Y., Ahmed, A., Beutel, A., Smola, A.J., Jing, H.: Recurrent recommender networks. In: *Proceedings of the tenth ACM International Conference on Web Search and Data Mining*, pp. 495–503. ACM (2017)
20. Xie, M., Hou, Y., Zhang, P., Li, J., Li, W., Song, D.: Modeling quantum entanglements in quantum language models (2015)
21. Zhang, P., Niu, J., Su, Z., Wang, B., Ma, L., Song, D.: End-to-end quantum-like language models with application to question answering (2018)
22. Zhang, Y., et al.: A quantum-inspired multimodal sentiment analysis framework. *Theor. Comput. Sci.* **752**, 21–40 (2018)
23. Zhang, Y., et al.: Sequential click prediction for sponsored search with recurrent neural networks. In: *Twenty-Eighth AAAI Conference on Artificial Intelligence*, pp. 1369–1375 (2014)
24. Zhou, G., et al.: Deep interest network for click-through rate prediction. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1059–1068. ACM (2018)
25. Zhu, Y., et al.: What to do next: modeling user behaviors by time-LSTM. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017*, pp. 3602–3608 (2017)
26. Zhu, Y., et al.: A brand-level ranking system with the customized attention-GRU model. In: *Proceedings of the Twenty-Seven International Joint Conference on Artificial Intelligence, IJCAI 2018* (2018)





# Multi-source Multi-net Micro-video Recommendation with Hidden Item Category Discovery

Jingwei Ma<sup>1</sup>, Jiahui Wen<sup>2</sup>, Mingyang Zhong<sup>1</sup>, Weitong Chen<sup>1</sup>(✉),  
Xiaofang Zhou<sup>1</sup>, and Jadwiga Indulska<sup>1</sup>

<sup>1</sup> School of ITEE, The University of Queensland, Brisbane, Australia  
{jingwei.ma,m.zhong1,w.chen9}@uq.edu.au,  
{zxf,jaga}@itee.uq.edu.au

<sup>2</sup> Academy of Military Sciences, Beijing, China  
wen\_jiahui@outlook.com

**Abstract.** As the sheer volume of available micro-videos often undermines the users' capability to choose the micro-videos, in this paper, we propose a multi-source multi-net micro-video recommendation model that recommends micro-videos fitting users' best interests. Different from existing works, as micro-video inherits the characteristics of social platforms, we simultaneously incorporate multi-source content data of items and multi-networks of users to learn user and item representations for recommendation. This information can be complementary to each other in a way that multi-modality data can bridge the semantic gap among items, while multi-type user networks, such as following and reposting, are able to propagate the preferences among users. Furthermore, to discover the hidden categories of micro-videos that properly match users' interests, we interactively learn the user-item representations. The resulted categorical representations are interacted with user representations to model user preferences at different level of hierarchies. Finally, multi-source content item data, multi-type user networks and hidden item categories are jointly modelled in a unified recommender, and the parameters of the model are collaboratively learned to boost the recommendation performance. Experiments on a real dataset demonstrate the effectiveness of the proposed model and its advantage over the state-of-the-art baselines.

**Keywords:** Micro-video recommendation · Multi-network modelling

## 1 Introduction

The explosion of micro-videos has arisen as a problem on social media in recent years, as the sheer volume of micro-videos can often undermine a users' capability to choose the micro-videos that best fit their interests. Recommender systems

---

J. Ma and J. Wen are contributed equally to this work.

© Springer Nature Switzerland AG 2019

G. Li et al. (Eds.): DASFAA 2019, LNCS 11447, pp. 384–400, 2019.

[https://doi.org/10.1007/978-3-030-18579-4\\_23](https://doi.org/10.1007/978-3-030-18579-4_23)

appear as a natural solution to this problem, helping social applications to efficiently determine the information offering to consumers and allowing users to quickly find the most useful information.

Most of the traditional methods for recommendation are based on Collaborative Filtering (CF) [17], however, that suffers from data sparseness and cold-start problem, as the only information they employed is the user-item interaction matrix that is extremely sparse. Thus, it is impossible to make recommendation for an unseen user or item unless its latent representation is learned beforehand. To approach this problem, many existing works incorporate additional information sources by joint learning users, items and auxiliary information such as texts, images, video and social connections. Although those methods that consider auxiliary information can be effective for recommendation task are not applicable for micro-video recommendation due to the following reasons. First, micro-videos involve multiple modalities such as textual, visual and acoustic, and each modality reveals different characteristic of micro-videos. For example, the descriptive text associated with a micro-video indicates the main content of the micro-video, while the frames in a micro-video carry the partial information of the content that are more intuitive than text. Therefore, it is necessary to design a recommender system that can cater for different characteristics of auxiliary information and incorporate high-level representations of modality information into a unified model for micro-video recommendation. In [5, 29], it only leverage data of a single modality, and they usually design a specific process of information extraction from the single-source data and tightly couple it with the recommend frameworks, which limits the scalability of the frameworks, such as integrating heterogeneous data sources. Second, many previous works employ social connections for improving recommendation accuracy based on the idea that social friends interrelated with strong social connections are more likely to have similar interests. However, many of them [15, 21] simply utilize the following relationship as social regularization, while ignoring other user relationships. In the previous research [1], information from a single social network may conflict with the true reasons underlying user-item interactions, while clues from multiple networks can be complementary to each other for accurate recommendation. For example, family members may be “friends” in a social network but with completely different item preferences, while users consume the same item explicitly indicating the similarity of their interests.

To address those problems, we propose a multi-source multi-net method for micro-video recommendation. Beside user-item interactions, we leverage heterogeneous information sources for better item profiling, and integrate the latent representations of the multiple sources into a unified model to promote personalized recommendation. Different domain information can be complementary to each other and reveals the true factors of user preferences over items.

Furthermore, we employ multiple social networks for propagating user representations in a shared latent space based on the idea that strong social ties are supposed to be located in a close proximity. We learn the hidden representations of users and items in an end-to-end neural network, and back-propagate the

recommendation errors to update the parameters of the network jointly. Thus the user and item representations can best explain the user preferences over the items by considering heterogeneous data sources and multiple social networks. The contributions of this paper are listed as follows.

- The proposed model incorporates multiple information sources and multiple user networks in a unified model for micro-video recommendation. Each modality can be complementary to each other for better modelling user and item representations.
- The proposed model discovers hidden item category information for the recommendation task. The hierarchy of item-level and category-level information can reflect the underlying reason of user preferences over items.
- We demonstrate the effectiveness of the proposed solution using a real dataset, and present insights and its advantage over the state-of-the-art recommenders with comprehensive experiments and analysis.

## 2 Related Work

Many existing video-oriented websites, such as YouTube<sup>1</sup>, MSN Video<sup>2</sup>, have provided video recommendation services. For example, content-based filtering approach [18] is employed in Youtube, where videos are recommended to users based on the past viewed videos. In VideoReach [17], video recommendation is performed based on the multi-modality relevance and users' click-through data.

The main limitation of these methods is that they only consider the video-video relations, while ignoring the related users' preferences.

In the past decade, deep learning techniques have been applied in several fields such as computer vision, speech recognition and natural language processing. As for recommendation, many previous works have tried to combine various neural network structures with collaborative filtering to boost the performance [11]. For example, [7] and [16] combine generalized matrix factorization with multi-layer perceptron to capture the interrelations between users and items. Some others [12, 23] apply auto-encoders to model user-item interactions, and the recommendations are made by reconstructing the user preferences over the item with the pre-trained de-noising auto-encoders. Although these are demonstrated to be effective in previous works, they are inapplicable in our scenario as the models are learned solely based on the user-item interactions.

Recently, many works propose to jointly model the auxiliary information associated with users or items for recommendation. The underlying reason of incorporating additional data sources is that they are highly inter-related with user-item interactions and can alleviate the data sparseness and cold-start problem. The considered data modalities include text, audio and visual. Those observable attributes are processed into abstract representations with popular deep learning techniques (e.g. CNNs, RNNs), and interact them with user or item

<sup>1</sup> <https://www.youtube.com/?hl=zh-cn>.

<sup>2</sup> <https://www.msn.com/en-us/video>.

representations to compensate sparse user-item interactions. However, most of those methods linearly combine representations from different sources, without considering the hierarchy of items and item categories for better modelling user-item interactions. Another limitation of those methods is that the underlying user networks that interlink the user have not been explicitly exploited. The projection of users into low-dimensional representations needs to preserve local structures in the user networks. [21] considers social connections and explicitly regularize social friends to have similar representations. However, those works leverage a single user network to propagate user similarities, and discovering similar users with information of single modality may provide conflict evidence and compromise recommendation accuracy [3, 19].

### 3 The Proposed Model

This section describes the details of the proposed model. Subsection 3.1 provides the problem formulation. After that, Subjects. 3.2 and 3.5 presents the modelling of content of micro-videos and the modelling of user networks that considers multiple-types of user networks, respectively. Multi-source data contents are utilized in our model, such as textual, visual features of micro-videos and user networks. Subsection 3.3 details the modelling of hidden category that learns the hidden categories of micro-videos that match users' interests. Subsection 3.6 describes the unified model that incorporates multiple data sources.

#### 3.1 Problem Formulation

We denote a user-item interaction matrix as  $\mathbf{R} \in \mathbb{R}^{M \times N}$ , where  $M$  and  $N$  denote the number of users and items respectively. The non-empty entries  $\mathbf{R}_{ij}$  refers to the positive interaction between user  $u_i \in \mathbb{U}$  and item  $v_j \in \mathbb{V}$ . In our case, each item  $v_j$  is associated with a textual description and a keyframe  $(\mathbf{x}_j, \mathbf{g}_j)$ . More importantly, there are multiple user networks, representing different relations such as following and liking. For a user network  $G^k = (\mathbb{U}, \mathbb{E}^k)$ , where  $\mathbb{U}, \mathbb{E}^k$  is the set of nodes and edges,  $(u_i, u_l) \in \mathbb{E}^k$  means there is a positive connection between  $u_i$  and  $u_l$ . In this work, our framework considers following and liking user networks as examples, which are easily extended to incorporate other user networks such as reposting. Given the interaction matrix  $\mathbf{R}$ , the set of user  $\mathbb{U}$  and item  $\mathbb{V}$ , the observable texts and images  $(\mathbf{x}_j, \mathbf{g}_j, j = 1, \dots, N)$  associated with the items and the user network  $G^k, k = 1, 2$ , our task is to predict the missing values in  $\mathbf{R}$ .

#### 3.2 Content Modelling

In this subsection, we describe the modelling of content information (i.e. textual and visual information) for the micro-videos. Since similar items are more likely to have similar textual descriptions and visual information, the latent representations of those items are supposed to be in a proximity close to each other in

the shared latent space. Therefore, the content information is able to bridge the semantic gaps between items, and we can learn better item latent representations by exploiting content information.

**Textual Representations.** The descriptive text  $\mathbf{x}_j$  associated with micro-video  $v_j$  summarizes the overall content of the item, and underlying reason of modelling textual data sources is that a user’s preference over an item can be explained by the fact that the user is attracted by the overall content of a micro-video. First, we transform each text,  $\mathbf{x}_j = \{w_j^n\}_{n=1}^{|\mathbf{x}_j|}$  where  $w_j^n$  is the  $n^{th}$  word in  $\mathbf{x}_j$ , into embedding vectors with Glove:  $\mathbf{E}_j = \{\mathbf{e}_j^n\}_{n=1}^{|\mathbf{x}_j|}$ . We pad the embedding vectors into the same length by padding 0 to the end of those word sequences. The embedding vectors are then fed into a convolution layer and a max pooling layer to obtain the representation of each document:

$$\mathbf{o}_j = CNN - maxpooling(\mathbf{E}_j) \quad (1)$$

where  $\mathbf{o}_j \in \mathbb{R}^{n1}$  is the output of the CNN and max-pooling layer. More details of this process are referred to [26]. The vector  $\mathbf{o}_j$  is then pass to a fully connected layer to obtain the hidden representation for the descriptive text  $\mathbf{x}_j$ , as shown in Eq. (2):

$$\mathbf{h}_j^{text} = f(\mathbf{W}_{text} \times \mathbf{o}_j + \mathbf{b}_{text}) \quad (2)$$

where  $\mathbf{W}_{text} \in \mathbb{R}^{n1 \times d}$  and  $\mathbf{b}_{text} \in \mathbb{R}^d$  are the transformation matrix and the biases, and  $n1$  is the size of the vocabulary and  $d$  is the predefined hidden size.

**Visual Representations.** The idea of extracting visual features from video frames for micro-video recommendation is that frames can reflect the user specific interests in a straightforward way, which are usually difficult to be described by text [22].

In this paper, we choose CNNs for extracting visual features, as CNNs are powerful in learning high-level visual representations for image classification and object detection. In the experiment, we select the first frame of each micro video as the visual feature, as the collected micro-video is short, 6 s, and keyframe can be used for long videos. We utilize the VGG-16 to obtain the visual features of video frames. For each micro-video frame  $\mathbf{g}_j$ , we use the output of the third-to-last layer, and pass it to a fully connected layer to obtain the representation of the frame, as shown in Eq. (3):

$$\mathbf{h}_j^{image} = f(\mathbf{W}_{image} \times CNN(\mathbf{g}_j) + \mathbf{b}_{image}) \quad (3)$$

where  $CNN(\mathbf{g}_j) \in \mathbb{R}^{4096}$  is the output of the third-to-last layer of the VGG-16 model.

**Latent-Content Modelling.** The idea of modelling item content is to learn mapping function to project item latent representations and content representations (i.e.  $\mathbf{h}_j^{text}$ ,  $\mathbf{h}_j^{image}$ ) of different modalities into a common space so that

the similarities between them can be directly measured. We denote the latent representation of item  $v_j$  as  $\mathbf{v}_j$ , and then the conditional probability of observing an item latent representation given its content representations as follows,

$$\begin{aligned}
 P(\mathbf{h}_j^{text}|\mathbf{v}_j) &= \frac{1}{1 + e^{-\mathbf{v}_j^\top \mathbf{M}^{text} \mathbf{h}_j^{text}}} \\
 &= \sigma(\mathbf{v}_j^\top \mathbf{M}^{text} \mathbf{h}_j^{text}) \\
 P(\mathbf{h}_j^{image}|\mathbf{v}_j) &= \frac{1}{1 + e^{-\mathbf{v}_j^\top \mathbf{M}^{image} \mathbf{h}_j^{image}}} \\
 &= \sigma(\mathbf{v}_j^\top \mathbf{M}^{image} \mathbf{h}_j^{image})
 \end{aligned} \tag{4}$$

where  $\mathbf{M}^{text}, \mathbf{M}^{image}$  are two linear transformation matrices that map an item latent representation and its content representations into a common space. The basic idea of Eq. (4) is that the latent representation of an item is similar to its content representation in the shared common space. The employment of sigmoid function for measuring the similarity between two objects is commonly used in previous works [2, 24].

### 3.3 Hidden Category Modelling

Existing works [4, 10] have demonstrated that information of different hierarchies can be incorporated to boost recommendation performance. For example, the category information of an item can help to capture user preferences in a more general granularity. While a user’s interest in a specific item is unclear, his/her general taste on the item category is obvious. Therefore, category and item representations can be complementary with each other for better modelling user preferences. However, the category information in our case is not explicitly available, so we propose to discover category-level information for the items, and model user preferences on both item- and category-levels.

Specifically, we perform clustering over the item representations, and results in several cluster centroids. Instead of regarding the centroids as category representations, we draw the category representations from a Gaussian distribution parameterized by the centroids. Thus the category representations can be used for modelling user preference in a more general granularity, which is detailed in the next subsection. Notice that the clustering process is iteratively performed with the item representations learning process, and they mutually benefit each other. As items consumed by similar users share similar characteristics, the learning of item representations drives similar items to have similar representations, and it benefits the clustering process. Furthermore, the category information can help to reveal real user preferences, which in return benefits the learning of representative item latent feature vectors.

In this paper, we employ KMeans for discovering latent categories and denote the category id for item  $v_j$  as  $c_j$ . The clustering process aims to minimize the following objects.

$$\operatorname{argmin} \sum_{k=1}^K \sum_{c_j=k} \|\mathbf{v}_j - \boldsymbol{\nu}_k\|^2 \quad (5)$$

where  $K$  is the pre-specified category number, and  $\boldsymbol{\nu}_k$  is the centroid for category  $k$ . The parameters  $\{c_j\}_{j=1}^N$  and  $\{\boldsymbol{\nu}_k\}_{k=1}^K$  is iteratively updated as in Eq. (6).

$$\begin{aligned} c_j &= \operatorname{argmin}_k \|\mathbf{v}_j - \boldsymbol{\nu}_k\|^2 \\ \boldsymbol{\nu}_k &= \frac{\sum_{c_j=k} \mathbf{v}_j}{\sum_j I(c_j = k)} \end{aligned} \quad (6)$$

where  $I(x) = 1$  if  $x$  holds, and 0 otherwise. Instead of using  $\{\boldsymbol{\nu}_k\}_{k=1}^K$  as the category representations, we introduce extra vectors  $\{\boldsymbol{\theta}_k\}_{k=1}^K$  for representing the categories, and assume Gaussian distribution on the residual noise of the clustered centroids as

$$P(\boldsymbol{\nu}_k | \boldsymbol{\theta}_k) = \mathcal{N}(\boldsymbol{\nu}_k | \boldsymbol{\theta}_k, \sigma_c^2 \mathbf{I}) \mathcal{N}(\boldsymbol{\theta}_k | 0, \sigma_\theta^2 \mathbf{I}) \quad (7)$$

where  $\mathcal{N}(x | \mu, \sigma^2)$  is the Gaussian distribution with mean  $\mu$  and variance  $\sigma^2$ , and  $\mathbf{I}$  is the identity matrix.  $\mathcal{N}(\boldsymbol{\theta}_k | 0, \sigma_\theta^2 \mathbf{I})$  is the Gaussian prior we place on the category representations to avoid overfitting.

### 3.4 Interaction Modelling

In this subsection, we introduce the modelling of user-item interaction. We choose Probabilistic Matrix Factorization (PMF) as our basic model for micro-video recommendation, as it is one of the most popular collaborative filtering models and is commonly used for recommendation tasks [20]. In this work, we have hierarchy of item- and category-level representations, and to capture user preferences at different level of granularity, we assume Gaussian distribution over observed interaction data as,

$$\begin{aligned} P(\mathbf{R}_{ij} | \mathbf{u}_i, \mathbf{v}_j, \boldsymbol{\theta}_{c_j}) &= \mathcal{N}(\mathbf{u}_i | 0, \sigma_u \mathbf{I}) \mathcal{N}(\mathbf{v}_j | 0, \sigma_v^2 \mathbf{I}) \\ &= \mathcal{N}(\mathbf{R}_{ij} | \mathbf{u}_i^T (\mathbf{v}_j + \boldsymbol{\theta}_{c_j}), \sigma^2 \mathbf{I}) \end{aligned} \quad (8)$$

where  $\mathcal{N}(\mathbf{u}_i | 0, \sigma_u \mathbf{I})$ ,  $\mathcal{N}(\mathbf{v}_j | 0, \sigma_v^2 \mathbf{I})$  are the Gaussian priors we place on the user and item representations  $\mathbf{u}_i$ ,  $\mathbf{v}_j$  respectively. In Eq. (8),  $\mathbf{u}_i^T \boldsymbol{\theta}_{c_j}$  explicitly model user's general preferences over the item categories, and  $\mathbf{u}_i^T \mathbf{v}_j$  model user preferences over the item content, as item representations are collaborative modelled with the raw content information in the common latent space (shown in Eq. (4)). Therefore, we model user interests with a hierarchy of item- and category-level information, which can be complementary with each other for promoting recommendation performance.

### 3.5 User Network Modelling

In social recommendation, the behaviour of a user is affected by his social neighbours, hence user preferences can be propagated through the social ties and encourage users with strong social connections to have the similar interests. Following [8], we employ probabilistic social matrix factorization to propagate preferences among social users. The advantage of our model is that we utilize multiple user networks to accurately model user's interests. We denote  $\mathbf{u}_i^k$  as the representation of user  $u_i$  in  $k^{th}$  user network. A user's representation depends on the latent representations of the connected social friends.

$$\mathbf{u}_i^k = \sum_{u_n^k \in N_i^k} T_{in}^k \mathbf{u}_n^k \quad (9)$$

where  $N_i^k$  is set of social ties of the user  $u_i$  in  $k^{th}$  user network, and  $T_{in}^k = \frac{1}{N_i^k}$ . The above equation implies that the representation of a user is the average of the representations of his/her connected neighbours. Similarly, the unified representation of a user across the user networks is dependent on the representations of the user in each network. We formulate the unified representation  $\mathbf{u}_i$  as the weighted sum of the representations  $\mathbf{u}_i^k$  across the networks.

$$\mathbf{u}_i = \sum_{k=1}^K \delta(\pi_{ik}) \mathbf{u}_i^k \quad (10)$$

where  $\delta(\pi_{ik})$  is a softmax function in the form of  $\delta(\pi_{ik}) = \frac{\exp(\pi_{ik})}{\sum_k \exp(\pi_{ik})}$ .  $\delta(\pi_{ik})$  can be explained as the impact of user representation in each individual network on the unified user representation. Considering the conditional probability of unified user representations, we have:

$$\begin{aligned} & P(\mathbf{u}_i, \{\mathbf{u}_i^k\}_{k=1}^K | \{T^k\}_{k=1}^K, \sigma_{U_1}^2, \sigma_{U_2}^2, \sigma_T^2) \\ & \propto P(\mathbf{u}_i | \{\mathbf{u}_i^k\}_{k=1}^K, \sigma_{U_1}^2, \mathbf{I}) \prod_k (P(\mathbf{u}_i^k | 0, \sigma_{U_2}^2, \mathbf{I}) \times P(\mathbf{u}_i^k | T^k, \sigma_T^2, \mathbf{I})) \\ & = \mathcal{N}(\mathbf{u}_i | \sum_{k=1}^K \delta(\pi_{ik}) \mathbf{u}_i^k, \sigma_{U_1}^2, \mathbf{I}) \\ & \times \prod_{k=1}^K (\mathcal{N}(\mathbf{u}_i^k | 0, \sigma_{U_2}^2, \mathbf{I}) \times \mathcal{N}(\mathbf{u}_i^k | \sum_{u_n^k \in N_i^k} T_{in}^k \mathbf{u}_n^k, \sigma_T^2, \mathbf{I})) \end{aligned} \quad (11)$$



### 3.6 The Unified Model

With the aforementioned information modelling, the conditional probability of the latent parameters given the observed data can be modelled as follows,

$$\begin{aligned}
 & P(\mathbb{U}, \mathbb{V}, \{\boldsymbol{\theta}\}_{k=1}^K | \mathbf{R}, \{\mathbf{x}_j\}_{j=1}^N, \{\mathbf{g}_j\}_{j=1}^N, \{G^k\}_{k=1}^2, \{\boldsymbol{\nu}_k\}_{k=1}^K) \\
 & \propto P(\mathbf{R} | \mathbb{U}, \mathbb{V}, \{\boldsymbol{\theta}\}_{k=1}^K) P(G | \mathbb{U}) P(\{\mathbf{x}_j\}_{j=1}^N | \mathbb{V}) P(\{\mathbf{g}_j\}_{j=1}^N | \mathbb{V}) \\
 & P(\{\boldsymbol{\nu}_k\}_{k=1}^K | \{\boldsymbol{\theta}_k\}_{k=1}^K) P(\mathbb{U}) P(\mathbb{V}) P(\{\boldsymbol{\theta}\}_{k=1}^K) \\
 & = \prod_{(u_i, v_j)} \{ \mathcal{N}(\mathbf{R}_{ij} | \mathbf{u}_i^T (\mathbf{v}_j + \boldsymbol{\theta}_{c_j}), \sigma^2 \mathbf{I}) \mathcal{N}(\mathbf{u}_i | \sum_{k=1}^K \delta(\pi_{ik}) \mathbf{u}_i^k, \sigma_{U_1}^2 \mathbf{I}) \\
 & \times \prod_{k=1}^K [\mathcal{N}(\mathbf{u}_i^k | 0, \sigma_{U_2}^2 \mathbf{I}) \mathcal{N}(\mathbf{u}_i^k | \sum_{u_n^k \in N_i^k} T_{in}^k \mathbf{u}_n^k, \sigma_T^2 \mathbf{I})] \\
 & \times \sigma (\mathbf{v}_j^T \mathbf{M}^{text} \mathbf{h}_j^{text}) \sigma (\mathbf{v}_j^T \mathbf{M}^{image} \mathbf{h}_j^{image}) \\
 & \times \mathcal{N}(\boldsymbol{\nu}_{c_j} | \boldsymbol{\theta}_{c_j}, \sigma_c^2 \mathbf{I}) \mathcal{N}(\boldsymbol{\theta}_{c_j} | 0, \sigma_\theta^2 \mathbf{I}) \mathcal{N}(\mathbf{u}_i | 0, \sigma_u \mathbf{I}) \mathcal{N}(\mathbf{v}_j | 0, \sigma_v^2 \mathbf{I}) \} \tag{12}
 \end{aligned}$$

By taking the negative likelihood of the above conditional probability, the loss function that needs to be minimized as follows,

$$\begin{aligned}
 \dagger \mathcal{L} &= \sum_{(u_i, v_j)} \{ \frac{1}{2} [\mathbf{R}_{ij} - \mathbf{u}_i^T (\mathbf{v}_j + \boldsymbol{\theta}_{c_j})]^2 \\
 & + \sum_k [\frac{\lambda_T}{2} (\mathbf{u}_i^k - \sum_{u_n^k \in N_i^k} T_{in}^k \mathbf{u}_n^k)^2 + \frac{\lambda_{U_2}}{2} (\mathbf{u}_i^k)^2] \\
 & + \frac{\lambda_{U_1}}{2} (\mathbf{u}_i - \sum_k \delta(\pi_{ik}) \mathbf{u}_i^k)^2 \\
 & - \alpha \log \sigma (\mathbf{v}_j^T \mathbf{M}^{text} \mathbf{h}_j^{text}) - \alpha \log \sigma (\mathbf{v}_j^T \mathbf{M}^{image} \mathbf{h}_j^{image}) \\
 & + \frac{\lambda_c}{2} (\boldsymbol{\nu}_{c_j} - \boldsymbol{\theta}_{c_j})^2 + \frac{\lambda_\theta}{2} (\boldsymbol{\theta}_{c_j})^2 + \frac{\lambda_u}{2} (\mathbf{u}_i)^2 + \frac{\lambda_v}{2} (\mathbf{v}_j)^2 \} \tag{13}
 \end{aligned}$$

where  $\lambda_T = \sigma_T^2 / \sigma^2$ ,  $\lambda_{U_1} = \sigma_{U_1}^2 / \sigma^2$ ,  $\lambda_{U_2} = \sigma_{U_2}^2 / \sigma^2$ ,  $\lambda_c = \sigma_c^2 / \sigma^2$ ,  $\lambda_\theta = \sigma_\theta^2 / \sigma^2$ ,  $\lambda_u = \sigma_u^2 / \sigma^2$ ,  $\lambda_v = \sigma_v^2 / \sigma^2$ ,  $\alpha = 2\sigma^2$ .

The objective function can be optimized by stochastic gradient descent (SGD) [9]. The update equations are as follows,

$$\boldsymbol{\theta}^{new} \leftarrow \boldsymbol{\theta}^{old} - lr \frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{14}$$

where  $lr$  is the learning rate, while the  $\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$  can be computed by chain rule in back-propagation.

## 4 Evaluation

### 4.1 Dataset

As the existing datasets (video datasets) either do not contain the social information or do not fit for micro-video scenarios, to validate the effectiveness of the proposed model, a real dataset is collected for our experiments. We collected a year’s data during 2015 and 2016 from Vine<sup>3</sup>. Inspired by the existing works such as [13, 14], we filter out the users with fewer than 5 review records and the micro-videos with fewer than 5 viewers. Finally, the processed dataset includes 9412 users, 19058 micro-videos and 109433 interactions. On average, each user has 11.6 records and each micro-video has 5.7 viewers.

### 4.2 Setup

We set the hyper-parameters of the proposed model to the following default values: for the regularization terms in the loss function, the  $\lambda$ s are the hyper-parameters, defaulted to 0.01; for the texts associated with the items, we initialize the embedding matrix with Glove and tune it during the training process; for parameters in CNNs processing the texts, we set the number of filters to be 100 for each of the filter size in the range of 2 to 4; for the training configuration, we set the initial learning rate to 0.001, and decay it by 0.95 for every 1000 steps. The batch size is set to 1024, and the model is trained for a maximum of 1000 epochs. For each user, we randomly sample 5 missing interactions as negative samples for each positive samples. In addition, for each user 70% of the respective positive and negative samples are used for training. Beside the following social network, another social network is created by connecting users if they like the same micro-videos. In addition to the aforementioned default values, we also evaluate the proposed model with different network structures.

As for the validation process, for each ground truth item for a user, we mix each test ground truth with 100 randomly sampled items, and rank the ground truth among the randomly-sampled items. Therefore, for a user with 5 positive items in the testing set, those items are rank with 500 unobserved items. The final result is an average over all users. Finally, we apply three commonly used metrics, including precision@k, recall@k and nDCG@k, to evaluate our model from different point of views.

### 4.3 Baselines

We compare the proposed model with the following state-of-the-art baselines.

- NeuMF [7]: applies matrix factorization with an end-to-end neural network.
- PACE [25]: introduces contexts to bridge the semantic gap between users and items, in addition to the modelling of user-item interactions.

---

<sup>3</sup> Vine: <https://vine.co/>.

- TrustSVD [6]: learns user and item representations from user-item interaction data. In addition, an adjacency matrix of trust network is leveraged to factorize truster and trustee representations.
- DeepCoNN [29]: utilizes user- and item-generated texts to represent users and items, and applies a deep neural network to model their interactions.
- CKE [27]: applies deep learning methods (stacked de-noising auto-encoders and stacked convolution auto-encoders) to joint learn item textual and visual representations for recommendation.
- JRL [28]: learns user and item representations in each individual information source (e.g. review text, image, numerical rating), that are then integrated to obtain the joint representations for users and items.
- MSN (the proposed model): incorporates multiple data sources and multiple user networks for micro-video recommendation with item category discovery.

We select these baselines as they constitute the state-of-the-art methods that consider information of different sources. For example, NeuMF models latent user and item representations from rating information with deep neural network, and DeepCoNN models the interaction between user-item pairs with the similarity measurement between their respective textual features. PACE and TrustSVD both consider user interaction and social information, and the difference is the way that they exploit social information for regularizing user representations. CKE and JRL incorporates data source of multiple modalities, and the different is that CKE extracts item features in each data source (e.g. text, image) and then generates unified item representations for modelling user-item interaction data, while JRL learns user and item representations in each data space and then combines them for recommendation.

### 4.4 Performance Comparison

The performance comparison among those models on different metrics are presented in Fig. 1. From the figures, we have the following observations.

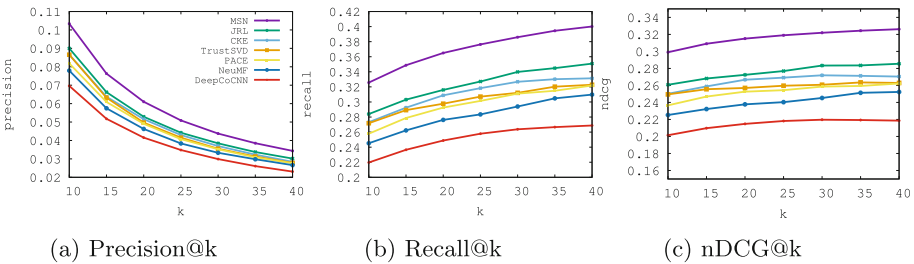


Fig. 1. Performance comparisons

First, we find that JRL performs better than CKE across different metrics, which has been demonstrated by previous work [28]. This is because JRL model

user-item representations and adds up their interaction scores from each of the information sources to rerank the top-N recommendations. Therefore, items can be ranked higher in the final recommendation lists as long as the user preferences over the items are properly profiled in any one of the information sources. On the contrary, CKE linearly combines the item representations over the information sources, and assume the unified item representations to be drawn from a Gaussian distribution parameterized by the linear combinations. As a result, evidence from different sources can negatively affect each other for reranking recommendation lists.

Second, the reason that JRL and CKE outperform DeepCoCNN with large margin is that the DeepCoCNN only consider the textual information for modelling the user-item interactions, while JRL and CKE include multiple data sources (e.g. interactions, texts and images) to learn user-item representations for the recommendation.

Third, PACE performs slightly better than NeuMF across different metrics, which demonstrates the benefit of incorporating social context for propagating user preferences. However, its improvement over NeuMF is marginal, and the reason is two-fold. The original PACE integrates geographic contexts for regularizing items, but they are not available in our dataset, and this component is ignored when implementing PACE. Moreover, we fine-tune the parameters for NeuMF as the original model faces the overfitting problem in our dataset due to the data sparseness. To do this, we add l2-norm for all the parameters and apply batch normalization on the output of fully-connect layers, and we finally grid search the learning rate to achieve the best result.

Also, TrustSVD outperforms PACE with large margin, especially on recall and nDCG. PACE leverages social contexts to regularize user latent vectors. The user latent vectors are used to predict their social contexts and the losses are back-propagated to update the vectors so that users who share the similar social friends are forced to be in a proximity that is close to each other. However, the regularization of users is detached from the objective function, namely the user latent features are not updated in a way towards the optimization of the recommendation performance. On the contrary, TrustSVD jointly optimizes the objective function and regularize user latent vectors together. More importantly, TrustSVD explicitly regularizes users with matrix factorization, while PACE implicitly propagates user preferences based on similar contexts, which may account for its insufficiency in learning real user preferences.

NeuMF uses deep neural networks to model user-item interactions. The advantage of the model is that the non-linear transformations in the neural network are able to capture informative user-item semantics for the recommendation, and the informative user-item semantics cannot be modelled with simple dot product in traditional matrix factorization. Moreover, due to the limited information sources, NeuMF still faces the problem of data sparseness, and this other models such as CKE and JRL significantly outperforms NeuMF, as they are able to incorporate rich information from both users and items.

Unexpectedly, DeepCoCNN performs the worst of all baselines. The model exploits user-generated and item-generated texts for modelling their respective latent features. The basic idea is that user-generated texts reflect users preference and texts associated with items indicate their characteristics. However, the features learned from texts are basically biased to the textual semantic and cannot be generalized to reflect user latent vectors. One possible solution to this problem is to add user and item embedding along with the textual features, and this result can be regarded as a variant of JRL.

Furthermore, we can conclude that JRL and CKE perform better than TrustSVD and PACE. However, these two types of recommenders are not comparable, as JRL and CKE leverage content information for modelling item representations, while TrustSVD and PACE exploit user network for propagating user preferences. For datasets where the social information is discriminative, TrustSVD and PACE may outperform JRL and CKE. And for datasets where content data is informative, JRL and CKE may be superior.

Finally, the proposed model MSN outperforms the state-of-the-art baselines with large margins. The advantage of the proposed over JRL and CKE demonstrates the effectiveness of incorporating user networks for recommendations. The information from multiple user networks can be exploited for better user profiling and network local structure preservation. The superiority of MSN over PACE and TrustSVD shows the benefit of extracting content information from item side for better item representations learning, as the data of different modalities associated with items can bridge similar items in the shared latent space. However, it may be unfair to compare with those baselines since we are able to integrate different data modalities for joint modelling user-item interactions, even though this is one of the major contributions of this paper. For a fair comparison, we examine the effectiveness of the proposed model with each modality (e.g. content, category and user network), when compared with the baselines.

#### 4.5 Structure Study

In this subsection, we study different variants of the proposed model to investigate the effect of each modelling component (e.g. content, category and user networks). The variants of our model are listed as follows:

- MSN-con: incorporates item content information and user-item interaction data for modelling user-item representations and recommendation.
- MSN-cat: models user-item interactions and discovers item category iteratively, and then incorporates category representations for predicting the rating scores.
- MSN-net: integrates user-item interaction and user networks for regularizing user representations and recommendation simultaneously.

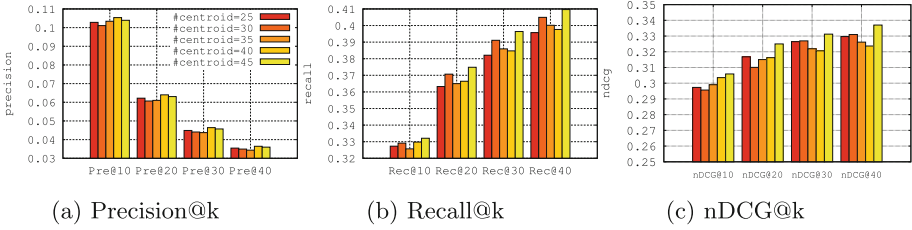
We compare MSN-con with JRL, CKE and DeepCoCNN since all of them explore item content information for the recommendation. MSN-cat is compared

with NeuMF, as both of them only consider rating matrix for user-item representations learning and recommendation, the difference is that MSN-cat dynamically discovers item category with clustering technique and incorporate the hierarchy of item- and category-level information for modelling user preference over the item at different granularities. Finally, MSN-net is compared with TrustSVD and PACE, because all of them take advantage of the user network for preserving the local structures in the network when learning user representations.

**Table 1.** Precision, Recall, nDCG@10 of different comparable models

Models	Precision@10	Recall@10	nDCG@10
DeepCoCNN	(0.06964)	(0.21963)	(0.20145)
CKE	(0.08656)	(0.27311)	(0.25021)
JRL	(0.08999)	(0.28397)	(0.26079)
MSN-con	(0.09076)	(0.28724)	(0.26327)
NeuMF	(0.07791)	(0.24517)	(0.22526)
MSN-cat	(0.0822)	(0.25879)	(0.23736)
PACE	(0.08183)	(0.25785)	(0.23659)
TrustSVD	(0.08654)	(0.27174)	(0.24948)
MSN-net	(0.09008)	(0.28384)	(0.26095)
MSN	(0.1034)	(0.3257)	(0.29904)

According to the comparison results presented in Table 1, we have the following observations. First, for models that leverage content information for recommendations, MSN-con, JRL and CKE achieve better performance than DeepCoCNN, this is because they exploit both textual and visual features for recommendations, while DeepCoCNN only considers textual information. The reason of performance variance among MSN-con, JRL and CKE is the way they employ to process the content data. We adopt convolution and max pooling mechanisms for processing texts and images, and they are proved to be able to capture the most informative features for information retrieval tasks [27]. CKE utilizes stacked de-noising auto-encoders for extracting textual and visual features, and JRL employs word embedding and convolution techniques to process texts and images. Even though the performance improvement of MSN-con is not noticeable, the proposed model MSN is able to outperform JRL and CKE significantly by incorporating user network and item category information. Furthermore, by comparing MSN-cat with NeuMF, we can observe the benefit of discovering latent item categories and incorporating them for modelling hierarchical user preferences. MSN-cat and NeuMF are comparable since both of them leverage user interaction data for learning user-item representations and recommendation, except that MSN-cat draws category representations from the centroids of clustered item representations and incorporates them to model user-item interactions. The hierarchy of item- and the category-level information is



**Fig. 2.** Recommendation performance as a function of the cluster centroids.

able to capture user interests at different levels of granularity and improve recommendation performance. Finally, the advantage of MSN-net over TrustSVD and PACE demonstrate the effectiveness of incorporating multiple user networks for propagating user preferences, since a single user network may contain conflicting evidence against the real factors underlying user-item interaction, while information from multiple user networks can be complementary to each other for better regularizing user representations. Comparing different variants of the proposed models, we can find that MSN-cat is inferior to MSN-con and MSN-net, this is because the only information available for MSN-cat is the rating matrix. Therefore, MSN-cat still faces the problem of data sparseness.

### 4.6 Parameter Study

In this paper, the number of centroids needs to be pre-specified for clustering items, thus we study the effect of the centroid number on the recommendation performance in this subsection. Few item clusters can make the category representations not discriminative enough to model hierarchical user preferences. While many centroids can make the category representations less informative for bridging items having similar characteristics, considering the special case where each item is regarded as a cluster. We present in Fig. 2 the recommendation performance across different metrics by varying the number of cluster centroids. We can see that the number of pre-specified cluster centroids have little impact on our recommendation model, as we are able to achieve similar recommendation performance when we vary the centroids number amongst. This is because with the different pre-specified number of centroids, we can cluster the items into different level of hierarchies. Therefore, as long as the categorical structure of the items data is preserved, the item categorical representations can be informative and discriminative for distinguishing items with different characteristics. The underlying reason that the discovered category representations can boost accurate recommendation performance is two-fold. From the items’ point of view, the discovered categorical information encourages items with similar characteristics to have a unified category representation, and the representation is able to bridge the semantic gap among items in the category. From the users’ perspective, by interacting users with both items and categories, we are able to capture user preferences at different levels of granularity, and an item tends to

be ranked higher in the recommendation list as long as the user preference on the item is properly profiled either on the item level or on the category level. This experiment demonstrates that it is flexible to specify the item category number for achieving competitive recommendation accuracy.

## 5 Conclusion

By leveraging the multi-modality information sources in micro-videos and the multi-type networks among users, we propose to incorporate the latent representations of the multiple sources into a unified model to facilitate recommendation, and employ multiple user networks for propagating user representations in a shared latent space. The multiple information sources act as bridges to interrelate items with similar content, while the user networks regularize users with strong social ties to have similar preferences. In addition, we propose to discover hidden categorical representations of micro-videos and interact them with user representations for boosting recommendation. The hidden categorical information can help to capture user preference at different levels of granularity. The modelling of hidden category and the user-item representations learning are iteratively performed in a unified model, and the recommendation losses are back-propagated to update the parameters in a way that can best optimize the recommendation performance. Finally, we validate the proposed model on a real dataset and study different variants of the proposed model, which demonstrates its advantage over the state-of-the-art baselines.

## References

1. Cao, C., Ge, H., Lu, H., Hu, X., Caverlee, J.: What are you known for?: Learning user topical profiles with implicit and explicit footprints. In: SIGIR (2017)
2. Chang, S., Han, W., Tang, J., Qi, G.J., Aggarwal, C.C., Huang, T.S.: Heterogeneous network embedding via deep architectures. In: KDD (2015)
3. Chen, W., Wang, S., Long, G., Yao, L., Sheng, Q.Z., Li, X.: Dynamic illness severity prediction via multi-task rnns for intensive care unit. In: ICDM (2018)
4. Chen, X., Qin, Z., Zhang, Y., Xu, T.: Learning to rank features for recommendation over multiple categories. In: SIGIR (2016)
5. Chen, X., Zhang, Y., Ai, Q., Xu, H., Yan, J., Qin, Z.: Personalized key frame recommendation. In: Proceedings of the 40th International ACM SIGIR (2017)
6. Guo, G., Zhang, J., Yorke-Smith, N.: TrustSVD: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings (2015)
7. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: WWW (2017)
8. Jamali, M., Ester, M.: A matrix factorization technique with trust propagation for recommendation in social networks. In: RecSys (2010)
9. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization (2014). arXiv preprint: [arXiv:1412.6980](https://arxiv.org/abs/1412.6980)
10. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD (2008)



11. Li, P., Wang, Z., Ren, Z., Bing, L., Lam, W.: Neural rating regression with abstractive tips generation for recommendation. In: SIGIR (2017)
12. Li, S., Kawale, J., Fu, Y.: Deep collaborative filtering via marginalized denoising auto-encoder. In: CIKM (2015)
13. Lian, D., Zhao, C., Xie, X., Sun, G., Chen, E., Rui, Y.: GeoMF: joint geographical modeling and matrix factorization for point-of-interest recommendation. In: KDD (2014)
14. Liang, D., Charlin, L., McInerney, J., Blei, D.M.: Modeling user exposure in recommendation. In: WWW (2016)
15. Ma, J., Li, G., Zhong, M., Zhao, X., Zhu, L., Li, X.: LGA: latent genre aware micro-video recommendation on social media. *MTAP* **77**(3), 2991–3008 (2018)
16. Manotumruksa, J., Macdonald, C., Ounis, I.: A deep recurrent collaborative filtering framework for venue recommendation. In: CIKM (2017)
17. Mei, T., Yang, B., Hua, X.S., Yang, L., Yang, S.Q., Li, S.: VideoReach: an online video recommendation system. In: SIGIR (2007)
18. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International Conference on World Wide Web, WWW 2001 (2001)
19. Wang, S., Chang, X., Li, X., Sheng, Q.Z., Chen, W.: Multi-task support vector machines for feature selection with shared knowledge discovery. *Signal Process.* **120**, 746–753 (2016)
20. Wang, S., Wang, Y., Tang, J., Shu, K., Ranganath, S., Liu, H.: What your images reveal: exploiting visual contents for point-of-interest recommendation. In: WWW (2017)
21. Wang, X., He, X., Nie, L., Chua, T.S.: Item silk road: recommending items from information domains to social users (2017). arXiv preprint: [arXiv:1706.03205](https://arxiv.org/abs/1706.03205)
22. Wen, J., Ma, J., Feng, Y., Zhong, M.: Hybrid attentive answer selection in CQA with deep users modelling. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
23. Wu, Y., DuBois, C., Zheng, A.X., Ester, M.: Collaborative denoising auto-encoders for top-n recommender systems. In: WSDM (2016)
24. Xu, L., Wei, X., Cao, J., Yu, P.S.: Embedding of embedding (EOE): joint embedding for coupled heterogeneous networks. In: WSDM (2017)
25. Yang, C., Bai, L., Zhang, C., Yuan, Q., Han, J.: Bridging collaborative filtering and semi-supervised learning: a neural approach for poi recommendation. In: KDD (2017)
26. Yu, P.S., Yu, P.S., Yu, P.S.: Joint deep modeling of users and items using reviews for recommendation. In: WSDM (2017)
27. Zhang, F., Yuan, N.J., Lian, D., Xie, X., Ma, W.Y.: Collaborative knowledge base embedding for recommender systems. In: KDD (2016)
28. Zhang, Y., Ai, Q., Chen, X., Croft, W.: Joint representation learning for top-n recommendation with heterogeneous information sources. In: CIKM (2017)
29. Zheng, L., Noroozi, V., Yu, P.S.: Joint deep modeling of users and items using reviews for recommendation. In: WSDM (2017)



# Incorporating Task-Oriented Representation in Text Classification

Xue Lei<sup>1</sup>, Yi Cai<sup>1</sup>(✉), Jingyun Xu<sup>1</sup>, Da Ren<sup>1</sup>, Qing Li<sup>2</sup>, and Ho-fung Leung<sup>3</sup>

<sup>1</sup> School of Software Engineering,  
South China University of Technology, Guangzhou, China  
lx.leixue@gmail.com, ycai@scut.edu.cn, jingyun.x@qq.com,  
edwardyam@outlook.com

<sup>2</sup> Department of Computing, The Hong Kong Polytechnic University,  
Hung Hom, Kowloon, Hong Kong  
qxili@polyu.edu.hk

<sup>3</sup> The Chinese University of Hong Kong, Hong Kong, Hong Kong SAR, China  
lhf@cuhk.edu.hk

**Abstract.** Text classification (TC) is an important task in natural language processing. Recently neural network has been applied to text classification and achieves significant improvement in performance. Since some documents are short and ambiguous, recent research enriches document representation with concepts of words extracted from an external knowledge base. However, this approach might incorporate task-irrelevant concepts or coarse granularity concepts that could not discriminate classes in a TC task. This might add noise to document representation and degrade TC performance. To tackle this problem, we propose a task-oriented representation that captures word-class relevance as task-relevant information. We integrate task-oriented representation in a CNN classification model to perform TC. Experimental results on widely used datasets show our approach outperforms comparison models.

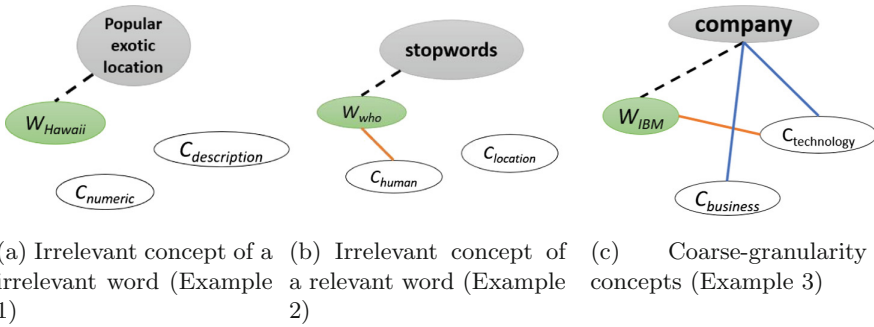
**Keywords:** Natural language processing · Text classification · Neural network

## 1 Introduction

Text Classification (TC) is an important task in natural language processing (NLP) and is widely applied to NLP tasks like information retrieval, web search, and sentiment analysis [2, 14, 16]. Recently, due to the efficacy of deep learning, neural network models like convolutional neural networks (CNN) and recurrent neural networks (RNN) are utilized to perform Text Classification [6]. An important part in text classification is document representation [6]. Generally, documents are represented by word embeddings and then document representations are fed to neural network models to perform classification. Some documents are short, ambiguous and sparse, which makes text classification hard. To resolve

**Table 1.** Example sentences of text classification. The correct class for each task is shown in bold. The first two examples are question type classification task. Classes in this task represent the type of answer to a question. e.g.,  $c_{numeric}$  means the answer to the question should be a numeric value.

Example	Sentence to be classified	Classes in a TC task
1	When did <i>Hawaii</i> become a state?	$c_{numeric}$ , $c_{description}$
2	<i>Who</i> died 1 feet from where John F. Kennedy did?	$c_{human}$ , $c_{location}$
3	<i>IBM</i> workers banned from using USB sticks	$c_{business}$ , $c_{technology}$



**Fig. 1.** Relation between words, classes and concepts of Table 1. Concepts of words extracted from Probase are shown as gray nodes. The white and green nodes represent classes and words, respectively. Dash lines between concept nodes and word nodes suggest they have ‘is-a’ relation. Red edges between word nodes and class nodes indicate the word is relevant to the class, which is drawn based on intuition. Blue edges represent the relevance between concepts and classes. (Color figure online)

this problem, Wang et al. [19] extract concepts of words from a knowledge base (KB), which is Probase [21]. They construct document representation which consists of word embeddings and concept embeddings (vectorial representation of concepts). Nevertheless, this approach has two limitations.

- (1) The first limitation is that incorporated concepts of words might be task-irrelevant. For this limitation, there could be two cases:
  - One case is that a word and the concept of this word both are irrelevant to the task. Let us consider Example 1.

**Example 1.** The TC task is to classify the first sentence in Table 1 into two classes:  $c_{numeric}$  and  $c_{description}$ .  $w$  and  $c$  with subscripts are used to denote words and classes respectively. This is a question type classification task. Intuitively,  $w_{hawaii}$  is irrelevant to both  $c_{numeric}$  and  $c_{description}$  in the text classification task. The concept of  $w_{hawaii}$  is “popular exotic location”, which is also irrelevant to both classes. As TC is performed based on classes in the task,  $w_{hawaii}$

can be considered as task-irrelevant. Incorporating the concept of  $w_{hawaii}$  might add noise to the document representation.

- The other case is that a word itself is task-relevant while the concept of the word is task-irrelevant. This can occur due to task-relevant concepts are not present in a knowledge base. Let us consider Example 2.

**Example 2.** The TC task is to classify the second sentence in Table 1 into  $c_{human}$  and  $c_{location}$ . This is also a question type classification task. We show the concept of  $w_{who}$  in Fig. 1(b), which is “stopwords”.  $w_{who}$  itself is relevant to  $c_{human}$  and it is a strong indicator of  $c_{human}$ . Yet “stopwords” is irrelevant to both  $c_{human}$  and  $c_{location}$ . Incorporating irrelevant concepts might introduce noise to document representation and lead to performance degradation.

- (2) The second limitation is that even though concepts incorporated are relevant, the granularity of these concepts might be too **coarse** to discriminate classes in the confronting TC task. Let us consider Example 3.

**Example 3.** The TC task is to classify the third sentence in Table 1 into  $c_{business}$  and  $c_{technology}$ . The sentence is a news headline. The concept of  $w_{IBM}$  is “company”, which is relevant to both  $c_{business}$  and  $c_{technology}$ . Thus, “company” might not discriminate these two classes. Incorporating this concept in document representation might not provide useful information to the confronting task.

If we could compute the relevance between words and classes, we could identify relevance between classes and documents that these words constituted. Since text classification is performed based on classes in the TC task, word-class relevance can be regarded as task-relevant information. Thus, we consider that it is necessary to incorporate word-class relevance in word representation. If word-class relevance between a word and each classes in the TC task are incorporated in representation of the word, the granularity of this representation is the same as classes. Thus the problem caused by incorporating coarse granularity can be alleviated.

In this paper, we propose an approach to incorporate task-oriented representation (TOR) that leverages task-relevant information in text classification. Specifically, we encode word-class relevance as task-relevant information in TOR and investigate methods to compute word-class relevance. Further, we study how to incorporate TOR into a CNN classification model.

The contributions of our work are summarized as follows:

- We propose an approach to incorporate task-relevant information in task-oriented representation to tackle the problem of including irrelevant concepts in word representation. Our approach also alleviates the problem caused by coarse-granularity concepts.
- Our proposed task-oriented representation is explicit and explainable while widely used word embeddings are implicit and not interpretable.
- We evaluate our proposed approach on widely used text classification datasets and our approach outperforms comparison approaches.

## 2 Related Work

Conventionally, one-hot representation of a document is used to perform Text Classification, where different term weighting schemes are designed to assign a score to each word [20]. Nevertheless, word order is not taken into consideration in those methods. Recently, due to the effectiveness of neural network, many neural network models (e.g., Convolutional Neural Network) are widely applied to perform Text Classification [6, 9, 10, 19, 23]. Some work seeks to design architecture of neural network model to better learn features from training data. Conneau et al. [1] build a very deep Convolutional Neural Networks. Zhou et al. [25] integrates bidirectional Long Short-Term Memory (LSTM) with Two-dimensional max pooling. [8] averages word embeddings in the same windows as input to the classification model. [17] introducing a 1-vs-res layer to tackle the problem of open space classification.

Some research focuses on incorporating linguistic features in neural network models. Johnson et al. [7] builds deep pyramid CNN on a word level. Wang et al. [5] build recursive neural model regarding the discourse dependency tree based on discourse structure. Li et al. [11] initialize CNN filters to encode semantic features of n-gram.

Besides, some study focus on enriching document representation by incorporating lexical resources. Xu et al. [22] utilize datasets of other languages and distill knowledge to the neural network model in a target language. Wang et al. [19] combine word-concept and character-level features as input. However, concept features from a general-purpose knowledge base might not be relevant to the classification task confronting. Also, these concepts might be too coarse with respect to the classification task. Therefore, in our work, we enrich input using discriminative information, in the sense that it is relevant to the classification task and free from granularity problem. In our work, we utilize the relevance between words and classes to enrich input representation as task-relevant information, which is captured by the proposed task-oriented representation of a word. There exists research focusing on the computation of relevance between words and classes [4]. However, these work adopts unsupervised approach, that is the class in while the class labels in our task are known. The computation of relevance between words and classes is supervised in our approach.

## 3 The Proposed Approach

The previous model incorporates concepts extracted from a KB to represent documents [19]. However, noise might be added to document representation by incorporating task-irrelevant concepts and coarse-granularity concepts, which results in performance degradation. To alleviate these problems, we leverage task-relevant information by encoding word-class relevance in the proposed task-oriented representation (TOR) of words. We introduce two methods to compute word-class relevance, which are class probability and word probability. Further, we present two possible methods to integrate TOR in a CNN classification model, which are concatenation and using subnetwork.

For notation, we denote words as  $w$ . Let  $C = \{c_1, c_2, \dots, c_{|C|}\}$  be a collection of classes in a particular text classification task and  $|C|$  is the total number of classes.  $\vec{v}_w$  and  $\vec{v}_w^r$  denote word embedding and task-oriented representation respectively.

### 3.1 Task-Oriented Representation

If we can obtain the word-class relevance between all words constituting a document and each classes in the TC task, we can identify the most relevant class of the document. To incorporate task-relevant information, we propose task-oriented representation (TOR) to capture word-class relevance. TOR is a vectorial representation of  $w$  denoted by  $\vec{v}^r$ . Each dimension of TOR represents the relevance between  $w$  and a class in the TC task. In this way, we encode word-class relevance between  $w$  with each class in a TC task. Formally, TOR of a word  $w$  is shown as follows:

$$\vec{v}_w^r = [r(w, c_1), r(w, c_2), \dots, r(w, c_{|C|})]$$

where  $r(w, c_i)$  is the relevance between  $w$  and  $c_i$  in the range of  $[0, 1]$ .

### 3.2 Word-Class Relevance Computation

In text classification, word-class relevance reflects how strongly connected a word ( $w$ ) and a class ( $c_i$ ) are. If  $w$  and  $c_i$  have a high relevance, then: (1) the occurrence of  $w$  in a document indicates a high probability that the document being labeled as  $c_i$ ; (2) conversely, given a  $c_i$  document,  $w$  appears in this document with a high probability. These two ideas can be mathematically modeled as  $p(c_i|w)$  and  $p(w|c_i)$ , which we name as **class probability** (CP) and **word probability** (WP), respectively. In what follows, we introduce how to estimate CP and WP in details.

**Class Probability.** The problem now is how to estimate  $p(c_i|w)$ . Intuitively, if  $w$  is frequently observed in documents of  $c_i$ ,  $p(c_i|w)$  should be high. We use  $|N|$  to denote word occurrence:  $|N_{w,c_i}|$  is the occurrence of  $w$  in  $c_i$  and  $|N_w|$  is the total occurrence of  $w$  in all classes. Then,  $p(c_i|w)$  can be estimated as:

$$\hat{p}(c_i|w) = \frac{|N_{w,c_i}|}{|N_w|}$$

The relevance between  $w$  and  $c_i$  computed using class probability is:

$$r_{CP}(w, c_i) = \hat{p}(c_i|w)$$

where  $\hat{p}(c_i|w)$  is the estimate of  $p(c_i|w)$ , which is used as the value of relevance between  $w$  and  $c_i$ . A drawback in CP is that the error in estimation of low-frequency word is relatively high. Let us consider Example 4.

**Example 4.** Suppose we are to classify documents into classes  $c_{politic}$  or  $c_{business}$ . The occurrence of a word  $w_{congress}$  is 25 in  $c_{politic}$  while the occurrence is 0 in  $c_{business}$ . Another word  $w_{cell}$  happens to appear once in  $c_{business}$ . Yet,  $\hat{p}(c|w)$  is the same for those two words, with  $\hat{p}(c_{politic}|w) = 1$  and  $\hat{p}(c_{business}|w) = 0$ . The error of estimation is lower in  $\hat{p}(c|w_{congress})$  than that in  $\hat{p}(c|w_{cell})$ . If  $r_{CP}(w, c_i)$  of words with high and low frequency are given as the same, the classification model could not differentiate between the TOR of these words. The task-relevant information encoded in TOR could not be utilized to find out the more informative words. This leads to performance degradation (Table 2).

**Table 2.** Example of  $p(c_i|w)$  estimate

$w$	$\hat{p}(c_{politic} w)$	$\hat{p}(c_{business} w)$
$w_{congress}$	1	0
$w_{cell}$	1	0

In a dataset, low-frequency words might constitute a large proportion of vocabulary. Therefore, we propose a strategy to tackle the problem caused by error in estimating CP of low-frequency word.

*K-Most Frequent.* We propose a strategy named K-Most Frequent (KMF) to tackle the problem of using CP to compute word-class relevance of low-frequency words. Since the error of  $\hat{p}(c_i|w)$  of low-frequency words are high, we only compute the word-class relevance of words with high frequency. Specifically, we use CP to initialize the top  $k$  most frequent words in the training set while randomly initialize TOR of other words. Thus, with KMF strategy the relevance between a word and each classes ( $r_{KMF}(w, c_i)$ ) is given as:

$$r_{KMF}(w, c_i) = \begin{cases} \hat{p}(c_i|w) & freq\_rank(w) < k \\ random & freq\_rank(w) > k \end{cases}$$

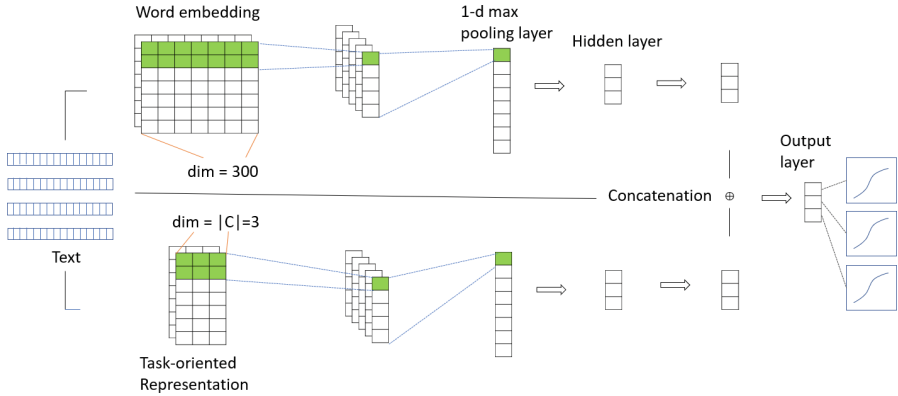
where  $freq\_rank(w)$  denotes the frequency rank of  $w$  with the most frequent word rank first.

**Word Probability.** WP differs from CP in that word frequency is considered. Let us revisit Example 4 for illustration. As  $w_{congress}$  occurs more often in  $c_{politic}$  than  $w_{cell}$ ,  $p(w_{congress}|c_{politic})$  is greater than  $p(w_{cell}|c_{politic})$ . This matches with our intuition that  $r(w_{congress}, c_{politic})$  should be higher than  $r(w_{cell}, c_{politic})$ . Formally, we use  $N_{c_i}$  to denote the total number of word occurrences in  $c_i$ . Thus, the estimate of  $p(w|c_i)$  follows:

$$\hat{p}(w|c_i) = \frac{|N_{w,c_i}|}{|N_{c_i}|}$$

Relevance between  $w$  and  $c_i$  computed using word probability is:

$$r_{WP}(w, c_i) = \hat{p}(w|c_i)$$



**Fig. 2.** Combining task-oriented representation and Word Embedding with two Sub-networks in triple classification

### 3.3 Integration of Task-Oriented Representation in Classification Model

In this section, we investigate methods to incorporate task-oriented representations to perform classification. Specifically, we introduce two possible methods which are concatenation and subnetwork (using two subnetworks to extract features from task-oriented representations and word embeddings simultaneously).

*Concatenation.* Intuitively, we can leverage task-oriented representations by concatenating task-oriented representations and word embeddings as input to a classification model. We employ the state-of-the-art text classification model (CNN), with filters that can capture n-gram features [9]. The input of the CNN model are  $\vec{v} \oplus \vec{v}^r$ , where  $\oplus$  denotes concatenation.

*Subnetwork.* The structure of vector space for word embedding and task-oriented representation might be different. While general semantic are encoded in word embedding space [15], we consider task-oriented information are encoded in vector space of TOR. Thus, inspired by Wang et al. [19], we apply two CNNs to encode features of word embeddings and TOR respectively. We combine features extracted from the two CNNs by concatenating intermediate hidden layers. The overall model is shown in Fig. 2. Specifically, the figure illustrates the classification model tackling a triple classification problem, where the dimension of TOR is three. The model consists of two components: the upper sub-network with word embeddings as input, and the lower sub-network with task-oriented representation as input. The two components are the same except for the input and each component consists of four layers: one input layer, one convolution layer, one pooling layer and one hidden layer. After concatenating the hidden layers of these two components, we apply an output layer on the concatenated vector to convert the output numbers into the probability of each class.



## 4 Experiments

### 4.1 Experiment Setup

We conduct experiments on commonly used datasets for text classification: TREC, AG News, and SST1 [12,18,24]. TREC is a question type classification dataset, which contains six different kinds of questions, such as questions about a person, a location, numeric information and description of something<sup>1</sup>. TREC is selected because many words in training documents are task-irrelevant and indiscriminative for the TC task. Our approach encodes task-relevant information in task-oriented representation. AG News is a dataset used in [19], which consists of news headline from four different classes. We choose AG News in order to evaluate our approach on a large dataset. SST1 is a fine-grained sentiment classification dataset consists of five classes, which are {very positive, positive, neural, negative, very negative}<sup>2</sup>. We evaluate our approach on SST1, which is different from the former two datasets in that: data of SST1 are classified by the degree of positiveness or negativeness, which can be considered as metric labeling problem [13]. We name CNN model that integrates TOR in TC as task-oriented CNN (TCNN) model. In Table 4, we show hyper parameters of our TCNN, tuned on dev set. We apply accuracy of prediction as evaluation metric. We use the 300-dimension word2vec pre-trained on Google News.<sup>3</sup> In the training process, Adagrad is used to optimize the loss function [3]. Experiments are conducted on an NVIDIA GTX1080Ti GPU.

**Table 3.** Dataset statistic

Datasets	#class ( $ C $ )	Training/Test set	Vocabulary size ( $ V $ )
TREC	6	5,452/500	9,170
AG News	4	120,000/7,600	32,309
SST1	5	11,855/2,210	16,112

**Table 4.** Hyper parameters of TCNN

Parameter	Values
Filter sizes	upper and lower: [2, 3, 4, 5]
Dropout rate	0.5
Hidden layers dimension	$ C $ (upper and lower)
Embedding dimension	300
Learning rate	$\alpha = 0.01$
Batch size	50

<sup>1</sup> <http://cogcomp.cs.illinois.edu/Data/QA/QC/>.

<sup>2</sup> <https://nlp.stanford.edu/sentiment/>.

<sup>3</sup> <https://code.google.com/archive/p/word2vec/>.

**Table 5.** Performance comparison of accuracy on TREC, AG and SST1

Model	TREC	AG	SST1
CNN-non-static	93.6	-	48.0
WCCNN [19]	91.2	85.6	-
KPCNN [19]	93.5	88.4	-
TCNN-CP-KMF (Subnetwork)	93.7	88.6	48.1
TCNN-WP (Subnetwork)	<b>94.0</b>	<b>89.0</b>	48.7
TCNN-CP-KMF (Concatenate)	93.3	87.3	48.0
TCNN-WP (Concatenate)	92.3	87.4	<b>49.5</b>

We compare our approach with several state-of-the-art approaches. (1) **CNN-non-static** This is proposed by Kim [9]. A CNN with a non-static channel is used to perform text classification. (2) **WCCNN** This is proposed by Wang et al. [19], where words and concepts obtained from Probase are used to enrich input representation. A CNN model with the static channel is used, following by a pooling layer to extract higher-level features. (3) **KPCNN** The methods combine word-level features used in WCCNN and character-level features to perform text classification. These two level features are extracted using two CNN separately and combined using a hidden layer. This method is also proposed by Wang et al. [19].

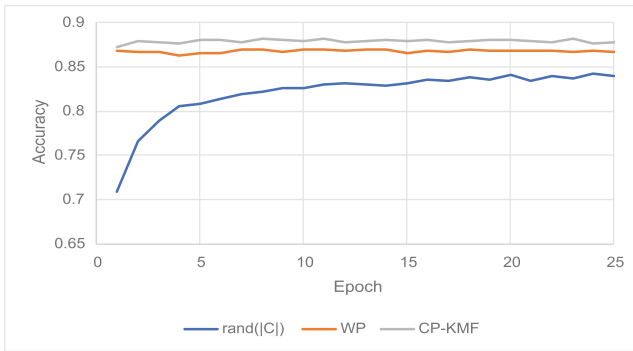
**Fig. 3.** Learning curves of  $\text{rand}(|C|d)$ , WP and CP-KMF on AG test set

Table 5 demonstrates results of our approach and comparison approaches on all datasets. CP-KMF stands for class probability with KMF strategy and WP is short for word probability. The parameter  $k$  is separately tuned on each dataset since the number of class-indicators in different datasets differs. We can observe that even without the external knowledge base, our approaches still outperforms KPCNN proposed by Wang et al. [19] which incorporates concepts

**Table 6.** Effectiveness of Task-oriented Representation (Class probability and Word probability)

Model	TREC	AG	SST1
Number of classes ( $ C $ )	6	4	5
$k$	$0.3 V $	$ V $	$0.4 V $
RAND ( $ C d$ )	86.1	85.5	35.6
RAND ( $100d$ )	88.2	85.3	37.9
RAND ( $200d$ )	89.7	84.7	35.7
RAND ( $300d$ )	88.9	84.2	37.1
CP ( $ C d$ )	78.7	87.6	35.2
CP-KMF ( $ C d$ )	<b>90.3</b>	<b>88.5</b>	40.3
WP ( $ C d$ )	90.2	86.3	<b>44.4</b>

from an external KB. The reason is that our proposed TOR incorporate task-relevant information rather than leveraging concepts of words which might be task-irrelevant or too coarse to be discriminative.

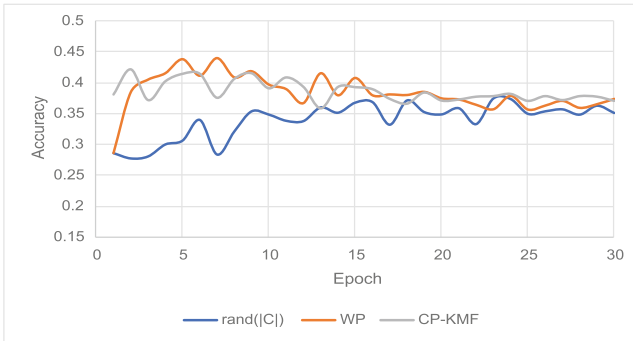
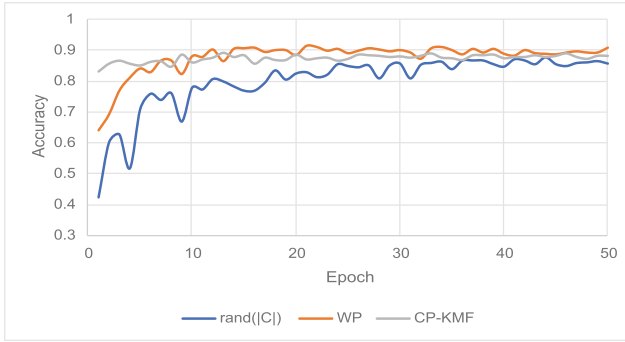
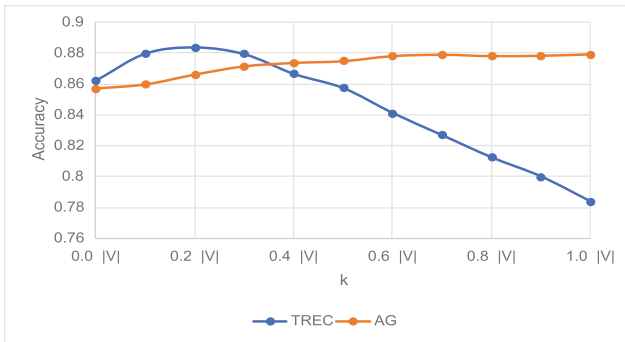
**Fig. 4.** Learning curves of  $\text{rand}(|C|d)$ , WP and CP-KMF on SST1 test set

Table 5 presents that word probability achieves the best results on all datasets. We consider it is due to WP leverages word frequency information, which is important in measuring word-class relevance. While CP-KMF also leverages word frequency information, it might still omit word frequency information of words which are randomly initialized. As shown in Table 5, concatenation performs inferior to using subnetworks on TREC and AG while outperforms using subnetworks on SST1. Additionally, using subnetwork significantly outperforms concatenation on AG (about 2%). The reason is there are more parameters in using subnetwork than concatenation and AG is larger than the other two datasets.



**Fig. 5.** Learning curves of  $\text{rand}(|C|d)$ , WP and CP-KMF on TREC test set

*Effect of  $k$  in Class Probability.* We investigate the effect of parameter  $k$  in CP-KMF on TREC and AG, as their size largely differ. We use  $|V|$  to denote the vocabulary size of a dataset. As shown in Fig. 6, we set  $k$  for each dataset to a percentage of the vocabulary size ( $|V|$ ) of the corresponding dataset. For example, with  $k$  set to  $0.1|V|$  and  $|V|$  of datasets shown in Fig. 3,  $k$  for TREC is 917 and  $k$  for AG is 3230.



**Fig. 6.** Learning curves of different  $k$  on TREC and AG dataset

Figure 6 shows that performance of TREC peaks for  $k = 0.3|V|$  and decreases afterwards. This suggests that the error in estimating class probability of low-frequency words hurt performance. There is no performance decrease as  $k$  increases on AG. We consider that the large size of AG makes it robust to noise introduced by error in estimating CP of low-frequency words.

*Effect of Task-oriented Representation.* To evaluate the efficacy of our proposed task-oriented representation, we compare TOR with a random baseline (Random vectors with the same dimension of TOR) as shown in Table 6. We use RAND

to indicate random vectors and show the dimension in following parentheses. Specifically, we only use the lower sub-network in Fig. 2, because we want to evaluate the effect of TOR. For a fair comparison, we set the dimension of the random vector the same as TOR such that the parameter in these two settings are the same, which means the capacity of two classification models is also the same. The input word embeddings of a classification model are often set between 50 and 500, therefore, we also perform experiments with a 100-dimension, 200-dimension and 300-dimension random vectors for comparison.

**Table 7.** Task-oriented representation of words ‘but’, ‘worst’ and ‘great’ before and after training

	Dimension	1	2	3	4	5
	Class	very positive	positive	neutral	negative	very negative
$\overrightarrow{v}_{but}^r$	Before training	0.2496	0.1771	0.1505	0.1723	0.2505
	After training	0.4532	0.1532	-0.0577	0.2111	0.4045
$\overrightarrow{v}_{worst}^r$	Before training	0.1270	0.1884	0.1953	0.2441	0.2451
	After training	-0.0946	0.0650	0.3512	0.4565	0.4714
$\overrightarrow{v}_{great}^r$	Before training	0.2236	0.2156	0.199	0.1835	0.1782
	After training	0.2980	0.2454	0.1994	0.1311	0.0701

As shown in Table 6, our approaches (CP-KMF and WP) outperform all random vectors (with dimension of  $|C|$ , 100, 200 and 300). This is noteworthy because the dimension of our input vector ( $\overrightarrow{v}^r$ ) is less than 10. Table 6 shows the specific dimension of TOR in the second line. However, the dimension of commonly used input vector is often about hundreds. For random input vector, limiting the dimension down to a small number might ( $|C|$ ) significantly degrade performance. This can be observed by comparing  $\text{RAND}(|C|d)$  with  $\text{RAND}(100d, 200d$  and  $300d)$ . Despite the relatively low dimension of our proposed task-oriented representation, TOR achieves a better result than the random vector with higher dimension. The performance improvement validates our assumption that task-relevant information is helpful to text classification.

We can see that CP-KMF outperforms CP without KMF. We think that the initial value of TOR computed with CP overfits the training dataset, as the CP is computed based on the statistic of the training dataset. It can be observed that CP without KMF performs badly on TREC. We hypothesize this is because the number of indicators of TREC is relatively small and a large proportion of words in vocabulary are noise. Thus using the class probability to compute word-class relevance degrades performance. CP without KMF perform better on AG than the other two dataset, this is due to the fact that AG is much larger than the other two datasets and classification model is able to generalize despite noise in initial value.

We show the learning curves of  $|C|$ -dimension random vector, TOR with WP and CP-KMF on TREC, AG and SST1 in Figs. 5, 3 and 4 respectively.

Generally, CP-KMF and WP converge faster than  $\text{rand}(|C|)$ . CP-KMF and WP achieve higher accuracy on first a few epochs than  $\text{rand}(|C|)$  except for WP on SST1. This encompasses our hypothesis that the task-relevant information encoded in TOR provides useful information to classification model.

## 5 Discussion

In this section, we compare task-oriented representation of words  $w_{but}$ ,  $w_{worst}$  and  $w_{great}$  before and after training on SST1 dataset. The TOR of these words are shown in Table 7. Since WP generally performs better than CP, the presented TOR is computed using WP. Table 7 shows the dimension of TOR and the corresponding classes in the first two lines. Every two rows in the chart represent TOR before and after training of a word. For the TOR after training, we color increased value in red and decreased value in green.

For TOR of  $w_{worst}$ , we can observe values ascend from the first to the last dimension. This suggests  $w_{worst}$  indicates negative sentiment, which validates that TOR incorporates task-relevant information. After training, the last three dimension of  $\vec{v}_{worst}^r$  increase while the first two decrease, which suggests classification model effectively captures information of training dataset. The change of value in TOR of the word  $w_{great}$  contrasts to  $w_{worst}$  because it is a positive sentiment word, which further validates that TOR captures task-relevant information.

$w_{but}$  is not a sentiment word yet sentiment expressed before and after ‘but’ is often different.  $\vec{v}_{but}^r$  shows a significant decrease in the third dimension and an increase in the first and last dimension after training. We consider this is because the occurrence of ‘but’ often indicates sentiment shift and emphasizes the sentiment after ‘but’. (e.g., ‘funny but overall limp’) Thus, ‘but’ is less likely to appears in neutral documents and the value of the third dimension decrease to  $-0.0577$ . As TOR is explicit and explainable, we could shed light on the behavior of the classification model.

## 6 Conclusion

In this paper, we propose an approach that incorporates task-oriented representation to perform text classification. Task-oriented representation of a word captures word-class relevance in a text classification. We present two different methods (class probability and word probability) to compute word-class relevance and further investigate two methods (concatenation and subnetwork) to integrate TOR in classification model.

**Acknowledgement.** This work was supported by the Fundamental Research Funds for the Central Universities, SCUT (No. 2017ZD048, D2182480), the Tiptop Scientific and Technical Innovative Youth Talents of Guangdong special support program (No. 2015-TQ01X633), the Science and Technology Planning Project of Guangdong Province (No. 2017B050506004), the Science and Technology Program of Guangzhou

International Science & Technology Cooperation Program (No. 201704030076). The research described in this paper has been supported by a collaborative research grant from the Hong Kong Research Grants Council (project no. C1031-18G).

## References

1. Conneau, A., Schwenk, H., Barrault, L., Lecun, Y.: Very deep convolutional networks for text classification. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, vol. 1, pp. 1107–1116 (2016). <https://doi.org/10.1007/s13218-012-0198-z>, <http://arxiv.org/abs/1606.01781>
2. Croft, W.B., Metzler, D., Strohman, T.: Search Engines: Information Retrieval in Practice, vol. 283. Addison-Wesley, Reading (2010)
3. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* **12**, 2121–2159 (2011). <https://doi.org/10.1109/CDC.2012.6426698>. <http://jmlr.org/papers/v12/duchi11a.html>
4. Hofmann, T.: Probabilistic latent semantic analysis. In: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, pp. 289–296. Morgan Kaufmann Publishers Inc. (1999)
5. Ji, Y., Smith, N.A.: Neural discourse structure for text categorization. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. Long Papers, vol. 1, pp. 996–1005 (2017)
6. Jin, P., Zhang, Y., Chen, X., Xia, Y.: Bag-of-embeddings for text classification. In: IJCAI International Joint Conference on Artificial Intelligence, vol. 16, pp. 2824–2830, January 2016. <https://www.ijcai.org/Proceedings/16/Papers/401.pdf>
7. Johnson, R., Zhang, T.: Deep pyramid convolutional neural networks for text categorization. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. Long Papers, vol. 1, pp. 562–570 (2017). <https://doi.org/10.18653/v1/P17-1052>, <http://aclweb.org/anthology/P17-1052>
8. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics. Short Papers, vol. 2, pp. 427–431 (2017)
9. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1746–1751 (2014)
10. Kottur, S., Moura, J.M.F., Lee, S., Batra, D.: Natural language does not emerge ‘Naturally’ in multi-agent dialog. *Synth. Lect. Hum. Lang. Technol.* **10**(1), 1–309 (2017). <https://doi.org/10.2200/S00762ED1V01Y201703HLT037>. <http://arxiv.org/abs/1706.08502>
11. Li, S., Zhao, Z., Liu, T., Hu, R., Du, X.: Initializing convolutional filters with semantic features for text classification. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 1884–1889 (2017)
12. Li, X., Roth, D.: Learning question classifiers. In: Proceedings of the 19th International Conference on Computational Linguistics, vol. 1, pp. 1–7. Association for Computational Linguistics (2002)
13. Pang, B., Lee, L.: Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pp. 115–124. Association for Computational Linguistics (2005)

14. Pang, B., Lee, L., et al.: Opinion mining and sentiment analysis. *Found. Trends® Inf. Retr.* **2**(1–2), 1–135 (2008)
15. Pennington, J., Socher, R., Manning, C.: Glove: global vectors for word representation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543 (2014)
16. Sehgal, U., Kaur, K., Kumar, P.: Notice of violation of IEEE publication principles the anatomy of a large-scale hyper textual web search engine. In: *Second International Conference on Computer and Electrical Engineering, ICCEE 2009*, vol. 2, pp. 491–495. IEEE (2009)
17. Shu, L., Xu, H., Liu, B.: Doc: deep open classification of text documents. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2911–2916 (2017)
18. Socher, R., Perelygin, A., Wu, J.: Recursive deep models for semantic compositionality over a sentiment treebank. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642 (2013). <https://www.aclweb.org/anthology/D13-1170>
19. Wang, J., Wang, Z., Zhang, D., Yan, J.: Combining knowledge with deep convolutional neural networks for short text classification. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pp. 2915–2921 (2017). <https://doi.org/10.24963/ijcai.2017/406>, <https://www.ijcai.org/proceedings/2017/406>
20. Wang, T., Cai, Y., Leung, H.f., Cai, Z., Min, H.: Entropy-based term weighting schemes for text categorization in VSM. In: *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 325–332. IEEE, November 2015. <https://doi.org/10.1109/ICTAI.2015.57>, <http://ieeexplore.ieee.org/document/7372153/>
21. Wu, W., Li, H., Wang, H., Zhu, K.Q.: Probase: a probabilistic taxonomy for text understanding. In: *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pp. 481–492. ACM (2012)
22. Xu, R., Yang, Y.: Cross-lingual distillation for text classification. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. Long Papers*, vol. 1, pp. 1415–1425 (2017)
23. Zhang, K., Zhu, K.Q., Hwang, S.W.: An association network for computing semantic relatedness. In: *AAAI*, pp. 593–600 (2015)
24. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: *Advances in Neural Information Processing Systems*, pp. 649–657 (2015)
25. Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H., Xu, B.: Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pp. 3485–3495 (2016)





# Music Playlist Recommendation with Long Short-Term Memory

Huiping Yang<sup>1</sup>, Yan Zhao<sup>1</sup>, Jinfu Xia<sup>1</sup>, Bin Yao<sup>2</sup>, Min Zhang<sup>1</sup>,  
and Kai Zheng<sup>3</sup>(✉)

<sup>1</sup> School of Computer Science and Technology, Soochow University, Suzhou, China  
{hpyang2017,jfxia}@stu.suda.edu.cn, {zhaoyan,minzhang}@suda.edu.cn

<sup>2</sup> Shanghai Jiao Tong University, Shanghai, China  
yaobin@cs.sjtu.edu.cn

<sup>3</sup> University of Electronic Science and Technology of China, Chengdu, China  
zhengkai@uestc.edu.cn

**Abstract.** Music playlist recommendation is an important component in modern music streaming services, which is used for improving user experience by regularly pushing personalized music playlists based on users' preferences. In this paper, we propose a novel music playlist recommendation problem, namely Personalized Music Playlist Recommendation (PMPR), which aims to provide a suitable playlist for a user by taking into account her long/short-term preferences and music contextual data. We propose a data-driven framework, which is comprised of two phases: *user/music feature extraction* and *music playlist recommendation*. In the first phase, we adopt a matrix factorization technique to obtain long-term features of users and songs, and utilize the Paragraph Vector (PV) approach, an advanced natural language processing technique, to capture music context features, which are the basis of the subsequent music playlist recommendation. In the second phase, we design two Attention-based Long Short-Term Memory (AB-LSTM) models, i.e., typical AB-LSTM model and Improved AB-LSTM (IAB-LSTM) model, to achieve the suitable personalized playlist recommendation. Finally, we conduct extensive experiments using a real-world dataset, verifying the practicability of our proposed methods.

## 1 Introduction

With the advent of lossy compression techniques (e.g., MP3 format), the field of music distribution has changed from being medium based to being digitized, which makes the music much easier to be downloaded or received by users on their personal computers and mobile devices via Internet. However, with massive amount of music available from thousands of web sites or online services, avoiding overwhelming choices and finding the “right” music have become a challenge for users. This is calling for effective music playlist recommendation techniques that can provide suitable music playlists for users.

Most existing studies focus on user-preference-based music playlist recommendation, which infer users' preferences from past music-listening patterns or

explicit feedbacks. For example, [7] characterizes both items and users by factor vectors inferred from item-rating patterns. However, they fail to effectively incorporate users' instant preferences and historical music-listening records. The overall music-listening behavior of a user may be determined by her long-term interest. But at any given time, a user is also affected by her instant preferences due to transient events, such as issuance of new songs in the current time.

In the bulk of playlist recommendation research, music content, which is primarily extracted from the audio signal, plays a key role in generating and recommending songs for users. For instance, Cano et al. [3] automatically extract descriptions related to instrumentation, rhythm and harmony from music audio signals and design a music browsing and recommendation system based on the high-level music audio data similarity. However, content-based music recommendation has not been applied very successfully in large range systems so far [6]. Music context data, referring to all music-relevant information that is not directly extractable from the audio signal itself, is another important factor for improving the quality of music recommendation. Context-based music recommendation approaches have higher user acceptance and even outperform content-based techniques for music retrieval [6, 17]. For example, Rendle et al. [14] explicitly model textual representations of musical knowledge (e.g., the pairwise interactions among users, items and tags) in music recommendation system, which performs well in runtime and achieves good recommendation quality. Cheng et al. [4] facilitate effective social music recommendation by considering users' location-related contexts as well as the global music popularity trends, which overcomes the cold-start and sparsity problems. Nevertheless, the work mentioned above ignores a crucial source of context-based data, comments of songs, which influence how a user (e.g., a listener) perceives music.

In this paper we propose a two-phase data-driven framework, namely Data-driven Music Playlist Recommendation (DMPR), which effectively combines users' long/short-term preferences and music contextual data. In the first phase, we obtain users' long-term preference features based on their favorite playlists, and songs' features (consisting of latent feature, semantic feature and category feature) based on their music context (i.e., lyrics, comments and belonging public playlists). In particular, we generate a rating matrix based on users' favorite playlists, and utilize a Matrix Factorization (MF) method to obtain users' long-term preference features for songs. The songs' latent features can be obtained by MF on the rating matrix as well. With the help of the Paragraph Vector (PV) approach [8], we can extract each song's semantic feature based on its lyrics and comments, and compute each song's category feature from its belonging public playlists. The second phase aims to combine a user's long/short-term preference features based on two Attention-based Long Short-Term Memory (AB-LSTM) models, i.e., typical AB-LSTM model and Improved AB-LSTM (IAB-LSTM) model, and recommend her a suitable playlist, which contains top- $k$  related songs that have the highest probability of being liked.

The contributions of this paper can be summarized as follows:

- (1) We provide a novel framework of Data-driven Music Playlist Recommendation (DMPR) based on users' long/short-term preferences and music contextual data, which aims to find a most suitable playlist for a user.
- (2) The Matrix Factorization technique is adopted to effectively extract users' long-term preference features and songs' latent features.
- (3) We introduce the Paragraph Vector (PV) approach, an advanced natural language processing technique, to extract the semantic features and category features of songs based on music context.
- (4) Two Attention-based Long Short-Term Memory (AB-LSTM) models, are designed to balance the long/short-term preferences of a user in order to find the most suitable music playlist for her.
- (5) We conduct extensive experiments on a real-world dataset, which empirically demonstrate the advantages of our proposed music playlist recommendation models compared to the baseline.

The remainder of this paper is organized as follows. Section 2 introduces the preliminary concepts and gives an overview of the proposed recommendation framework. Then we extract major features used in our work in Sect. 3. Two kinds of recommendation algorithms are presented in Sect. 4, followed by the experimental results presented in Sect. 5. Section 6 surveys the related works based on existing researches on music recommendation. Finally we conclude this paper in Sect. 7.

## 2 Problem Statement

In this section, we introduce some preliminary concepts and give an overview of the proposed recommendation framework. Table 1 summarizes the major notations used in the rest of the paper.

### 2.1 Preliminary Concept

**Definition 1 (Song).** A song, denoted by  $s = \langle l, c \rangle$ , consists of its lyrics  $s.l$  and comments  $s.c$ . In addition, we use  $S$  to represent a set of songs.

**Definition 2 (User).** A user, denoted by  $u = \langle f, h \rangle$ , consists of her favorite playlist  $u.f$  and historical playlist records  $u.h$ . In particular, the favorite playlist of user  $u$ , denoted by  $u.f = (s_1, s_2, \dots, s_n)$ , is a sequence of songs which have been marked as "like" by user  $u$ , and the playlist records history of user  $u$ , denoted by  $u.h = (s_1, s_2, \dots, s_m)$ , is a finite sequence of songs sorted by time when  $u$  heard the songs recently. We use  $U$  to represent a set of users.

**Definition 3 (Public Playlist).** A public playlist, denoted by  $pl = (s_1, s_2, \dots, s_n)$ , is a finite sequence of songs, which is created by the active users in the community.

**Table 1.** Summary of notations

Notation	Definition
$s$	A song
$s.l$	The lyrics of song $s$
$s.c$	The comments of song $s$
$S$	A song set
$u$	A user
$u.f$	The favorite playlist of user $u$
$u.h$	The historical playlist records of user $u$
$U$	A user set
$r_{u,s}$	Rating between user $u$ and song $s$
$pl$	A public playlist
$pl.v$	Vector representation of the public playlist $pl$
$e^U$	Latent preference feature matrix of user set $U$
$e^S$	Latent feature matrix of song set $S$
$s.v$	Distributed representation of song $s$
$s.v(l)$	Lyric vector representation of song $s$
$s.v(c)$	Comment vector representation of song $s$
$s.v(ca)$	Category vector representation of song $s$
$p^u$	Probability vector of user $u$

Note that the public playlist is different from albums, and everyone in the community can visit it without limits. For example, a HipHop fan can create a public playlist named “The Best HipHop 100”, which consists of 100 HipHop songs, and each user can access to this public playlist.

**Problem Statement.** Given a set of users, a set of songs and public playlists, our Personalized Music Playlist Recommendation (PMPR) problem aims to provide each user an ideal playlist, which contains the top- $k$  related songs that have the highest probability of being liked.

## 2.2 Framework Overview

The proposed framework shown in Fig. 1 consists of two major phases: (1) feature extraction; (2) music playlist recommendation.

**Feature Extraction.** This phase models long-term features of both users and songs. The features considered in our proposed Data-driven Music Playlist Recommendation (DMPR) framework can be divided into two major parts: (1) user preference feature, which describes a user’s long-term preference for music; (2) music features, which describe the music context information.

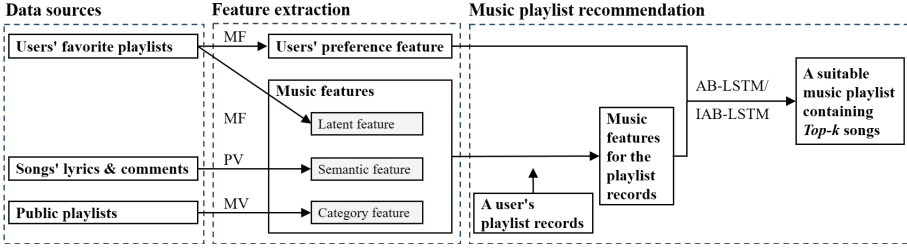


Fig. 1. Framework overview

User preference feature represents a user's long-term preference for music, which can be obtained by Matrix Factorization (MF) based on a user-song rating matrix, which is observed from users' favorite playlists. MF maps both users and songs into a joint latent space, thus the user-song rating matrix is modeled as inner products of a user feature matrix and a song feature matrix. The user feature matrix and the song feature matrix contain latent features in this latent space, where a user's latent feature is regarded as her preference for music.

Music features comprehensively describe a song's latent feature, as well as its lyrics, comments and category information. Specifically, the music features extracted in our work consist of three parts: latent feature, semantic feature of both lyrics and comments, and category feature. Latent features of songs, which implies the ratings between users and songs, can be captured from users' favorite playlists via MF. We use a Paragraph Vector (PV) approach to obtain semantic feature from each song's lyrics and comments and design a Mean Value (MV) method to extract each song's category feature based on the public playlists, which usually contain a sequence of similar songs.

**Music Playlist Recommendation.** In this phase, we use two Attention-based Long Short-Term Memory (AB-LSTM) models to recommend a user the top- $k$  related songs that have the highest probability of being liked based on her long/short-term preferences and songs' music features. The user preference feature extracted in the first phase represents a user's long-term preference for music, and the current songs' music features represent her short-term preference for music. Based on a user's long/short-term preferences for music, we propose two AB-LSTM model, i.e., typical AB-LSTM model and Improved AB-LSTM (IAB-LSTM) model, to recommend a suitable playlist to her.

### 3 Feature Extraction

In this section, we extract the main features that will be used in our work. The features can be mainly divided into two parts: (1) user preference feature, which describes a user's long-term preference for music; (2) music features, which describe the music context information.

### 3.1 User Preference Feature Extraction

In this section, we model users’ long-term preference feature by Matrix Factorization (MF). MF performs well in learning latent features of users and songs from the observed ratings in the user-song rating matrix. Therefore we utilize the MF technique to model users’ latent preference feature based on the user-song rating matrix, which can be obtained from users’ favorite playlists, in order to describe the users’ long-term preferences for songs.

We first generate a user-song rating matrix,  $R \in \mathbb{R}^{N \times M}$ , which consists of  $N$  users and  $M$  songs based on users’ favorite playlists. Each entry  $r_{u,s}$  in matrix  $R$  denotes user  $u$ ’s rating on song  $s$ . For instance, if song  $s$  exists in user  $u$ ’s favorite playlists, then we have an indication that user  $u$  likes song  $s$  (i.e.,  $r_{u,s} = 1$ ). Otherwise, we set  $r_{u,s} = 0$ . We use two latent feature matrices to represent users and songs respectively, namely user feature matrix ( $e^U \in \mathbb{R}^{N \times d}$ ) and song feature matrix ( $e^S \in \mathbb{R}^{M \times d}$ ), which explain the ratings between users and songs. MF maps ratings between users and songs into a latent space, such that users’ preference for songs is modeled as inner product between  $e^U$  and  $e^S$  in that latent space. The mapping of users’ latent preference feature matrix  $e^U$  and songs’ latent feature matrix  $e^S$ , is achieved by approximating the rating matrix by solving the following optimization problem:

$$\min_{e^U, e^S} \sum_{(u,s) \in K} (r_{u,s} - e_u^U e_s^{S^T})^2 + \lambda(\|e^U\|^2 + \|e^S\|^2), \tag{1}$$

where  $K$  is the set of <user, song> pairs observed from users’ favorite playlists,  $r_{u,s}$  is the rating between user  $u$  and song  $s$ ,  $e_u^U$  denotes the latent preference feature of user  $u$ ,  $e_s^S$  denotes the latent feature of song  $s$  and  $\lambda$  is the regularization coefficient. The regularization coefficient  $\lambda$  is used to avoid overfitting. We apply gradient descent algorithm to solve the optimization problem in Eq. 1 and obtain the users’ latent preference feature matrix  $e^U$  and songs’ latent feature matrix  $e^S$ .

### 3.2 Music Feature Extraction

Music features comprehensively describe each song’s latent feature, semantics and category. As shown in Fig. 2, the music features extracted in our work consist of three parts: (1) latent feature; (2) semantic feature of lyrics and comments; (3) category feature.

**Latent Feature Extraction.** Latent feature of song  $s$ , denoted as  $e_s^S$ , describes not only its musical identity but also many significant qualities that are relevant to understanding users’ musical preferences. With the user-song rating matrix generated from users’ favorite playlists, we can obtain each song’s latent feature based on MF, following the same process as of user preference feature.

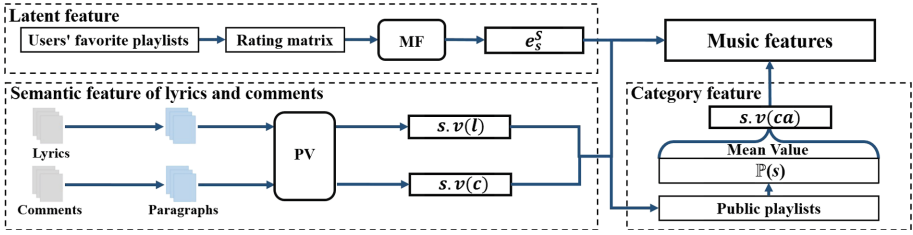


Fig. 2. Music feature extraction

**Semantic Feature Extraction.** In this part, we capture the songs’ semantic feature of lyrics and comments to describe their music context. Lyrics is an important aspect of musical semantics since they usually imply the information about the artist/performer, e.g., cultural background, political orientation, and style of music [6]. Comments in the community are user-generated content, which have an increasingly impact on a user’s preference.

We utilize the Paragraph Vector (PV) [8] technique, which is an unsupervised algorithm that learns continuous distributed vector representations for texts with any length, to obtain semantic feature of lyrics and comments. PV builds a word matrix  $W$ , where every word is mapped to a unique vector represented by a column, and builds a paragraph matrix  $D$ , where every paragraph is mapped to a unique vector represented by a column. For instance, a song’s comments are considered as a sequence of words in a paragraph, denoted as  $(w_1, w_2, \dots, w_T)$ . This comment paragraph is mapped into a unique vector represented by a column in matrix  $D$  and every word is mapped into a unique vector represented by a column in matrix  $W$ . In PV, the word vectors are asked to contribute to a prediction task about the next word, and the paragraph vector of this comment paragraph should also contribute to the prediction task of the next word when the contexts (sampled from the paragraph) are given. Thus, the goal of PV is to maximize the average log probability as follows:

$$\frac{1}{T} \sum_{t=k}^{T-k} \log p(w_t | w_{t-k}, \dots, w_{t+k}), \tag{2}$$

where  $T$  is the length of the current paragraph,  $k$  controls the size of context window, and  $p(w_t | w_{t-k}, \dots, w_{t+k})$  is the probability that the predicted word is word  $w_t$ . The prediction task for the predicted word can be done via a multi-class classifier like softmax, which can be computed as follows:

$$p(w_t | w_{t-k}, \dots, w_{t+k}) = \frac{\exp(y_{w_t})}{\sum_i \exp(y_i)}, \tag{3}$$

where  $\exp$  is the exponential function,  $y$  is a probability vector, and  $y_i$  is the un-normalized log-probability for word  $i$  to be the predicted word.  $y$  can be computed in Eq. 4.

$$y = b + Uh(w_{t-k}, \dots, w_{t+k}; W; D), \tag{4}$$

where  $U$  and  $b$  are the softmax parameters,  $h$  is constructed by a concatenation of the word vectors of  $(w_{t-k}, \dots, w_{t+k})$  and the paragraph vector extracted from word matrix  $W$  and paragraph matrix  $D$ .

After training, we get word matrix  $W$ , softmax weights  $U$  and  $b$ , and paragraph matrix  $D$  on comments and lyrics. Then we add new lyrics paragraphs and comment paragraphs as columns in  $D$  and use gradient descent on  $D$  while holding  $W$ ,  $U$ ,  $b$  fixed. Finally, we obtain the lyric vector representation  $s.v(l)$  and comment vector representation  $s.v(c)$  of song  $s$  from paragraph matrix  $D$ .

**Category Feature Extraction.** As mentioned before, a public playlist consists of a sequence of similar songs, which can be regarded as the same category. Therefore, we obtain the songs' category information based on public playlists.

With the lyrics vector representation  $s.v(l)$  and comment vector representation  $s.v(c)$  for each song  $s$  contained in the public playlist  $pl$ , the vector representation of  $pl$ , denoted by  $pl.v$ , can be formulated as follows:

$$pl.v = \frac{1}{n} \sum_{i=1}^n g(s_i.v(l), s_i.v(c)), \quad (5)$$

where  $n$  is the amount of songs in the public playlist  $pl$ ,  $g$  is constructed by a concatenation of  $s_i.v(l)$  and  $s_i.v(c)$ .

It is worth noting that a song may be contained in multiple public playlists, such that we calculate the song's category information by combining all its belonging public playlists' vectors as follows:

$$s.v(ca) = \frac{\sum_{pl \in \mathbb{P}(s)} pl.v}{|\mathbb{P}(s)|}, \quad (6)$$

where  $s.v(ca)$  is song  $s$ 's category vector representation,  $\mathbb{P}(s)$  is a set of public playlists that contain song  $s$  and  $|\mathbb{P}(s)|$  is the size of  $\mathbb{P}(s)$ . In addition, we concatenate semantic feature of lyrics  $s.v(l)$  and comments  $s.v(c)$ , and category feature  $s.v(ca)$  as the distributed representation of song  $s$ , denoted by  $s.v \in \mathbb{R}^k$ .

Given a song  $s$ , we can describe its music features by a combination of the latent feature  $e_s^S$ , and the distributed representation  $s.v$ .

## 4 Music Playlist Recommendation

In this section, we introduce our Attention-based Long Short-Term Memory (AB-LSTM) and Improved Attention-based Long Short-Term Memory (IAB-LSTM), which generate personalized playlists based on users' long/short preferences for songs and music context features.

LSTM is a variant of RNN, which is effective and scalable for sequential prediction problems [16]. Considering the time-ordered playlist records as sequential data, we adopt the Long Short-Term Memory (LSTM) to generate a suitable playlist that fits a user's musical interests. Recently, attention-based neural networks have been successfully used in many tasks like machine translation.



For example, [1] calculates weight attention score for each word in original sentences during the translation of the word. In our model, the attention mechanism can be used to calculate the weight attention scores for users' long/short-term preferences, which helps in recommendation performance. As a result, we use attention-based LSTM model to recommend suitable playlists for users. In particular, we design two models with different attention mechanisms. Firstly we adopt a typical Attention-Based Long Short-Term Memory (AB-LSTM) model to recommend playlists. Then we make some modification based on AB-LSTM, namely Improved Attention-Based Long Short-Term Memory (IAB-LSTM) model, to get better performance in recommendation.

### 4.1 AB-LSTM-Based Music Playlist Recommendation

In this section, we apply AB-LSTM model to recommend suitable songs for an individual user.

The architecture of AB-LSTM model is shown in Fig. 3, which contains three layers: input layer, hidden layer and output layer. The input layer contains the latent preference feature matrix of the user set  $U$  ( $e^U \in \mathbb{R}^{N \times d}$ ,  $N$  is the quantity of users in  $U$ ), and user  $u$ 's historical playlist records  $u.h = (s_1, \dots, s_t, \dots, s_m)$ . At each time step  $t$ , we concatenate the latent feature of song  $s_t$ , denoted by  $e_{s_t}^S \in \mathbb{R}^d$ , and the distributed representation of song  $s_t$ , denoted by  $s_t.v \in \mathbb{R}^k$ , as the music feature of song  $s_t$ , and input the music feature of song  $s_t$  and user preference feature matrix  $e^U$  to the hidden layer respectively.

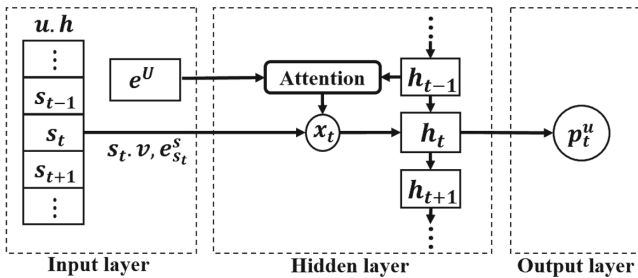


Fig. 3. Diagram of attention-based long short-term memory

The hidden layer, consisting of LSTM cells, is the key component of the AB-LSTM model, in which the hidden state can be computed as:

$$h_t = LSTM(h_{t-1}, x_t), \tag{7}$$

where  $h_t$  is the hidden state at time step  $t$  in the hidden layer, and  $x_t$  is a concatenation of the latent feature of song  $s_t$  ( $e_{s_t}^S$ ), song  $s_t$ 's distributed representation ( $s_t.v$ ) and the context vector of user  $u$  at time  $t$   $c_t^u$ . The context vector  $c_t^u$  is calculated in the attention part based on the user latent preference feature

and the hidden state in the previous time step, which can be considered as an extra input to help AB-LSTM to fully get long-term preference of the user.  $c_t^u$  is calculated in Eq. 8.

$$c_t^u = \sum_v^U a_{v,t}^u e_v^U, \quad (8)$$

where  $a_{v,t}^u$  is the attention weight of user  $u$  on user  $v$  at time step  $t$ ,  $e_v^U$  represents the latent preference feature of user  $v$ . The attention weight  $a_{v,t}^u$  is computed as follows:

$$a_{v,t}^u = \frac{\exp(\sigma(h_{t-1}, e_v^U))}{\sum_{v'}^U \exp(\sigma(h_{t-1}, e_{v'}^U))}, \quad (9)$$

where  $\sigma$  is a feed-forward neural network to produce a real-valued score. The attention weight  $a_{v,t}^u$  determines which user's latent preference feature should be selected to generate user  $u$ 's probability vector of predicted songs.

The probability vector of  $M$  predicted songs for user  $u$  at time step  $t$ , denoted as  $p_t^u \in \mathbb{R}^M$ , can be calculated by a single layer neural network activated by softmax in the output layer. The output of AB-LSTM model,  $p_t^u$ , can be computed as:

$$p_t^u = \text{softmax}(g(h_t)) = \text{softmax}(Wh_t + b), \quad (10)$$

where  $W$  and  $b$  are parameters of single layer neural network  $g$ , softmax function is used to squash the probability vector into a vector where each entry is in the range  $(0, 1)$ , and all the entries add up to 1. In this model, we usually use  $p_m^u$  at the last time step  $m$  as the probability vector of the predicted songs for user  $u$ , denoted by  $p^u$ .

Finally, we generate the top- $k$  related songs that have the highest probability of being liked by the user. As mentioned before, user  $u$ 's probability vector of the predicted songs, denoted by  $p^u$ , is a  $M$ -length vector, where  $M$  is the amount of predicted songs. The value of  $p_j^u$  corresponds to the predicted probability that user  $u$  likes the  $j$ -th predicted song. Given a specified playlist length  $k$ , we select the top- $k$  songs with the  $k$  greatest probability values from user  $u$ 's probability vector ( $p^u$ ) as the recommendation result.

## 4.2 IAB-LSTM-Based Music Playlist Recommendation

We further propose a novel model, namely Improved Attention-based Long Short-Term Memory (IAB-LSTM), with a new designed attention mechanism to recommend suitable songs for an individual user. We utilize an attention layer between hidden layer and output layer of IAB-LSTM, to replace the attention weight calculation during each time step of AB-LSTM's hidden layer, which helps in improving recommendation effectiveness. More importantly, IAB-LSTM focuses on calculating the attention weight of hidden states (implying the user's recent listened songs' music features) while AB-LSTM calculates the attention weight of user preference feature, which makes IAB-LSTM better in capturing the user's short-term preference.

The architecture of IAB-LSTM model is shown in Fig. 4, which contains four layers: input layer, hidden layer, attention layer and output layer. The

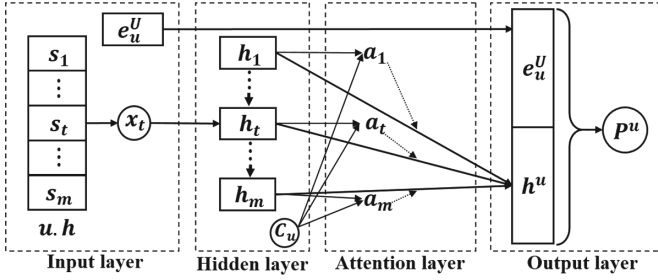


Fig. 4. Diagram of improved attention-based long short-term memory

input layer contains the latent preference feature of the user  $u$ , denoted by  $e_u^U \in \mathbb{R}^d$ , and her historical playlist records  $u.h = (s_1, \dots, s_t, \dots, s_m)$ . And at each time step  $t$ , we concatenate the latent feature of song  $s_t$ , denoted by  $e_{s_t}^S \in \mathbb{R}^d$ , and the distributed representation of song  $s_t$ , denoted by  $s_t.v \in \mathbb{R}^k$ , as an input  $x_t$  to the hidden layer. The hidden layer, consisting of LSTM cells, is an important component of the IAB-LSTM model, in which the hidden state can be computed as:

$$h_t = LSTM(h_{t-1}, x_t), \tag{11}$$

where  $h_t$  is the hidden state at time step  $t$  in the hidden layer.

The attention layer is the key component of IAB-LSTM, in which we summarize all hidden states as  $h^u$ , which is computed as follows:

$$h^u = \sum_{t=1}^m a_t^u h_t, \tag{12}$$

where  $m$  is the quantity of hidden states,  $a_t^u$  is the attention weight of user  $u$  on hidden state  $h_t$  at time step  $t$ , which is computed as follows:

$$a_t^u = \frac{\exp(g(h_t)^T C_u)}{\sum_{t'} \exp(g(h_{t'})^T C_u)}, g(h_t) = \tanh(W h_t + b), \tag{13}$$

where  $W$  and  $b$  are parameters of single layer neural network  $g$ , and context vector  $C_u$  is randomly initialized and jointly learned during the training process. The context vector  $C_u$  can be seen as a high level representation of a fixed query “what is the informative song” in the user  $u$ ’s historical playlist records.

Finally, the probability vector of  $M$  predicted songs for user  $u$ , denoted as  $p^u \in \mathbb{R}^M$ , can be calculated by a single layer neural network activated by softmax in the output layer. The output of IAB-LSTM model,  $p^u$ , can be computed as:

$$p^u = \text{softmax}(g'(h^u, e_u^U)) = \text{softmax}(W'(h^u \oplus e_u^U) + b'), \tag{14}$$

where  $p^u$  is user  $u$ ’s probability vector of the predicted songs,  $e_u^U$  is the latent preference feature of user  $u$ ,  $W'$  and  $b'$  are parameters of single layer neural network  $g'$ ,  $\oplus$  is defined as a concatenation and  $h^u$  is calculated in Eq. 12. Getting the probability vector, we generate a suitable playlist for the user by the same way as AB-LSTM.

## 5 Experiments

In this section, we conduct extensive experiments on a real-world dataset to study the performance of the proposed models. All the algorithms are implemented on an Intel(R) Xeon(R) CPU E7-4860 v2 @ 2.60 GHz with 64 G RAM.

### 5.1 Experiment Setup

**Dataset.** We use a real-world dataset generated by Netease Cloud Music, which is a freemium music streaming service. Netease Cloud Music has 300 million users and a music database consisting of more than over 10 million songs. Specifically, we crawl a dataset containing 35365 users, 1496 public playlists, 35469 songs and 377194 comments.

**Evaluation.** We study and compare the performance of the following algorithms:

- (1) CF: a model-based Collaborative Filtering approach [7], which utilizes matrix factorization to calculate the ratings between all the users and songs and recommends the top- $k$  songs with highest rating values for each user.
- (2) BPR: Bayesian Personalized Ranking [13] ranks each user’s preference for songs and provides a top- $k$  recommendation.
- (3) P-AB-LSTM: Preference-Based AB-LSTM, an AB-LSTM without song’s distributed representation (i.e., a song’s semantics and category information).
- (4) P-IAB-LSTM: Preference-Based IAB-LSTM without song’s distributed representation (i.e., a song’s semantics and category information).
- (5) AB-LSTM: Attention-Based Long Short-Term Memory model (based on both the user’s preference and song’s distributed representation).
- (6) IAB-LSTM: Improved Attention-Based Long Short-Term Memory model (based on both the user’s preference and song’s distributed representation).

Four widely-used metrics, Precision@ $k$  ( $P@k$ , the accuracy rate of top- $k$  recommendation), Normalized Discounted Cumulative Gain@ $k$  ( $NDCG@k$ ), Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR), are used to evaluate the recommendation precision of the above methods. The greater values of the above metrics mean the better performance. CPU time is given by the average time cost of recommending the top- $k$  songs by the recommendation models. We also evaluate the recommended playlists based on the number of recalled songs. Specifically, for each user, we use CF, AB-LSTM and IAB-LSTM to generate a playlist containing 10 songs and compare it with her real playlist records of next 10 songs. In addition, CF selects top-10 related songs with highest ratings as the generated playlist. A greater number of recalled songs mean better performance.

Table 2 shows our experimental settings, where the default values of all parameters are underlined.

**Table 2.** Experiment parameters

Parameters	Values
Size of ratings	25%, 50%, 75%, 100%
Length of historical playlist records	20, <u>30</u> , 40, 50
Amount of features extracted via MF	20, <u>40</u> , 60, 80

## 5.2 Experiment Results

We first compare our proposed models with two baseline methods including CF and BPR. The experimental results are summarized in Table 3. Our models consistently outperform the baseline methods by a noticeable margin. Taking the P@10 as an example, our IAB-LSTM achieves the most accurate recommendation result, whose accuracy is improved by around 83% compared with CF and 152% compared with BPR.

**Table 3.** Results of models

	P@3	P@5	P@10	NDCG@3	NDCG@5	NDCG@10	MAP	MRR
CF	0.0509	0.0517	0.0580	0.0846	0.1024	0.1376	0.1104	0.1181
BPR	0.0474	0.0453	0.0421	0.0856	0.1009	0.1150	0.0996	0.1077
P-AB-LSTM	0.0838	0.0822	0.0919	0.1047	0.1154	0.1742	0.1752	0.1920
P-IAB-LSTM	0.0891	0.0930	0.1020	0.1159	0.1440	0.2279	0.1840	0.2001
AB-LSTM	0.0817	0.0832	0.1061	0.1113	0.1281	0.1859	0.1858	0.2040
IAB-LSTM	<b>0.0962</b>	<b>0.0966</b>	<b>0.1356</b>	<b>0.1416</b>	<b>0.1640</b>	<b>0.2430</b>	<b>0.1916</b>	<b>0.2151</b>

**Effect of the Size of Ratings.** In this part of experiments, we change the size of the ratings used in feature extraction and study their effects on music playlist recommendation. As shown in Fig. 5, the precision of all approaches increases (with MAP and P@10 increase) when more ratings are used. Among all the methods, IAB-LSTM achieves the highest precision since IAB-LSTM effectively captures users' preferences and music context features, which demonstrates the robustness of our proposed algorithms. The CPU cost of all methods is not apparently affected by the size of the ratings since the prediction phrase is not directly computed from the ratings. Moreover, AB-LSTM (IAB-LSTM) runs slower than P-AB-LSTM (P-IAB-LSTM) because of the extra time cost for integrating the song's semantics and category features into the recommendation.

**Effect of the Length of Historical Playlist Records.** Figure 6 illustrates the effect of the length of historical playlist records on the performance of our models. Naturally the precision of all model gradually increases (with MAP and P@10 increase) as the length of historical playlist records grows. Obviously the performance of IAB-LSTM is better than P-IAB-LSTM, which proves the effectiveness of song's distributed representation. The CPU cost increases when the length of historical playlist records increases, since the size of input in our models is determined by the length of historical playlist records.

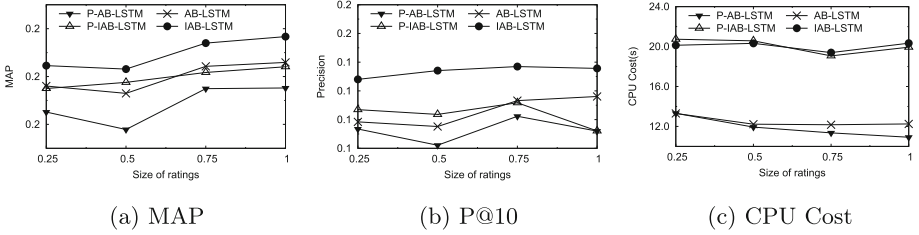


Fig. 5. Effect of the size of ratings

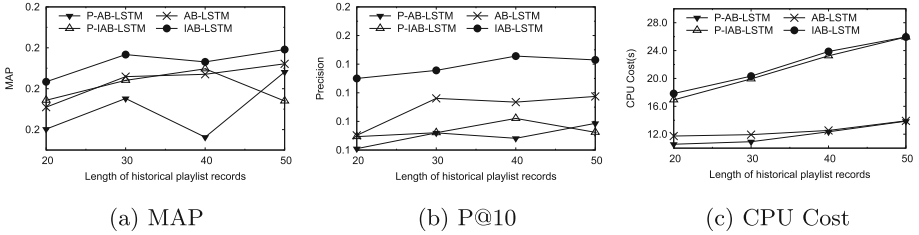


Fig. 6. Effect of the length of historical playlist records

**Effect of the Amount of Features Extracted via MF.** Next we study the effects of the amount of features extracted via MF. From Fig. 7 we can see that the precision of all algorithms increases (with MAP and P@10 increase) when the amount of features extracted via MF grows. The CPU cost slightly increases when the amount of features extracted via MF increases, since the greater size of the latent preference feature of the user increases computational complexity.

**Recalled Songs.** Finally we provide the recommended music playlists generated by some algorithms (i.e., CF, AB-LSTM and IAB-LSTM) for two users who are randomly selected from the dataset. The predicted songs existing in the user’s real playlist records will be marked by ✓ in Table 4, which shows that our proposed IAB-LSTM can better capture the preferences of users than other methods. Taking user 16596 as an example, IAB-LSTM recalls 4 songs and

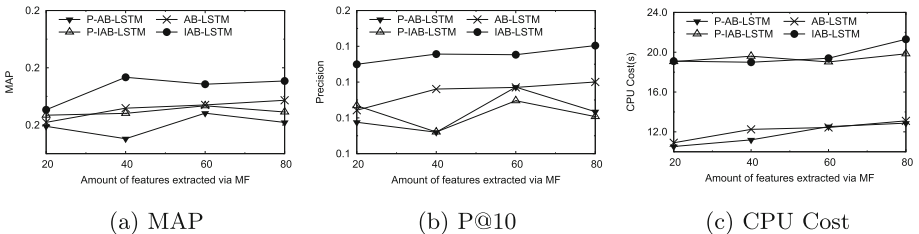


Fig. 7. Effect of the amount of features extracted via MF

AB-LSTM recalls 2 songs, while CF recalls 0 song. This means our model can make better music playlist recommendation with users' long/short term preferences and music context features.

**Table 4.** The recalled songs from Top-10 candidates

User Id	CF	AB-LSTM	IAB-LSTM
16596	We Don't Talk Anymore	Lie ✓	She Say
	Fade	Willow	Lie ✓
	Into A River	Iron Heart	I Am You
	Something	Old Street	In Fact, No
	Make You Mine	Travel	Along The Way
	A Half	Be What You Wanna Be	Cheng Du
	East Of Eden	Chasing The Wind	Outside The Light Years
	Es rappelt im Karton	Young And Young ✓	Lone Ranger ✓
	Counting Stars	Yesterday	Young And Young ✓
	I Am You	Star Falling	Trap ✓
8759	Say Honestly	Those Were The Days	Stupid ✓
	Special	Waste	Lie ✓
	Lover Boy 88	When You	Blue Lotus
	Jocelyn Flores	Lie ✓	Don't Understand
	Complete	Iron Heart	Vincent
	Regret	Young And Young	Don't Talk
	Meet	Yesterday	Young And Young
	Destination	Trap	Mind
	My Heart Will Go On	Calorie	Trap
	No Sad No Bad	Star Falling	Best

## 6 Related Work

The music playlist recommendation has attracted a number of researchers in recent years. In this section, we categorize the major methodologies used by recommendation systems as being based on: (1) music content; (2) music context.

**Music Content-Based Recommendation.** Music content-based features refer to tonality, pitch, and beat, symbolic features extracted from the music scores [9]. Existing research in the area of audio content-based music recommendation usually focuses on measuring music similarity [2]. For example, McFee et al. [10] treat music similarity learning as an information retrieval problem, where the music similarity is learned to optimize the ranked list of results in response to a query song. Dieleman et al. [11] use content-based latent factors to produce sensible recommendations, ignoring the fact that there is a large semantic gap between the song's characteristics and its corresponding audio signal. Wang et al. [19] simultaneously learn features from audio content and make personalized

recommendation, which performs well on both the warm-start and cold-start problems. However, content-based music recommendation has not been applied very successfully in large range systems so far [6].

**Music Context-Based Recommendation.** In music recommendation field, contextual data refers to all music-relevant information that is not included in the audio signal itself. In particular, we review three main types of context-based approaches: (1) text-retrieval-based approaches; (2) co-occurrence-based approaches; (3) user-preference-based approaches.

Text-retrieval-based approaches exploit textual representations of musical knowledge originating from web pages, user tags, or song lyrics. Oramas et al. [12] exploit tags and textual descriptions to extract and link entities to external graphs which are in turn used to semantically enrich the initial data in music recommendation. Schedl et al. [15] address the problem of similarity measurement among music artists via text-based features extracted from Web pages.

Co-occurrence-based approaches follow an immediate mechanism to estimate similarity based on the occurrence of two music pieces or artists within the same context like web pages, microblogs, playlists, and Peer-to-Peer (P2P) networks. Zangerle et al. [20] use the absolute numbers of co-occurrences between songs in order to measure the similarities between songs and artists, which helps in music recommendation systems.

User-preference-based approaches usually estimate music context similarity based on users' feedbacks. Cheng et al. [5] present a venue-aware music recommender system that recommends music to match different types of common venues in user's everyday life. Wang et al. [18] learn the low dimensional representations of music pieces from users' music listening sequences using neural network models. Cheng et al. [4] develop an effective social music recommendation system by considering users' location-related contexts as well as the global music popularity trends.

## 7 Conclusions

In this paper, we study the problem of Personalized Music Playlist Recommendation, where each user can receive a personalized music playlist based on her historical playlist records and music context. To settle this problem, we propose a novel Data-driven Music Playlist Recommendation (DMPR) framework, which incorporates long/short-term preferences of users and music features to improve the performance of recommendation. We address a few challenges by proposing different strategies to extract the long-term features of users and songs and designing effective AB-LSTM models to recommend a personalized music playlist (including top- $k$  related songs that have the highest probability of being liked) for each user by obtaining her short-term preference. Extensive empirical study based on a real dataset demonstrates our proposed models can effectively capture long/short-term preferences of users via attention mechanisms, and recommend suitable personalized playlists to users.



**Acknowledgement.** This work was supported by the NSFC (61832017, 61532018, 61836007, 61872235, 61729202, U1636210), and The National Key Research and Development Program of China (2018YFC1504504).

## References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. *CoRR* (2014)
2. Bogdanov, D., Haro, M., Fuhrmann, F., Xambó, A., Gómez, E., Herrera, P.: Semantic audio content-based music recommendation and visualization based on user preference examples. *IPM* **49**(1), 13–33 (2013)
3. Cano, P., Koppenberger, M., Wack, N.: Content-based music audio recommendation. In: *ACM Multimedia*, pp. 211–212 (2005)
4. Cheng, Z., Shen, J.: Just-for-me: an adaptive personalization system for location-aware social music recommendation. In: *ICMR*, p. 185 (2014)
5. Cheng, Z., Shen, J.: On effective location-aware music recommendation. *ACM TOIS* **34**(2), 13 (2016)
6. Knees, P., Schedl, M.: A survey of music similarity and recommendation from music context data. *TOMCCAP* **10**(1), 2 (2013)
7. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. *Computer* **42**(8), 30–37 (2009)
8. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: *ICML*, pp. 1188–1196 (2014)
9. Li, T., Ogihara, M., Li, Q.: A comparative study on content-based music genre classification. In: *SIGIR*, pp. 282–289 (2003)
10. McFee, B., Barrington, L., Lanckriet, G.: Learning content similarity for music recommendation. *TASLP* **20**(8), 2207–2218 (2012)
11. Van den Oord, A., Dieleman, S., Schrauwen, B.: Deep content-based music recommendation. In: *NIPS*, pp. 2643–2651 (2013)
12. Oramas, S., Ostuni, V.C., Noia, T.D., Serra, X., Sciascio, E.D.: Sound and music recommendation with knowledge graphs. *ACM TIST* **8**(2), 21 (2017)
13. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: *AUAI*, pp. 452–461 (2009)
14. Rendle, S., Schmidt-Thieme, L.: Pairwise interaction tensor factorization for personalized tag recommendation. In: *WSDM*, pp. 81–90 (2010)
15. Schedl, M., Pohle, T., Knees, P., Widmer, G.: Exploring the music similarity space on the web. *ACM TOIS* **29**(3), 14 (2011)
16. Schmidhuber, J., Wierstra, D., Gomez, F.J.: Evolino: hybrid neuroevolution/optimal linear search for sequence prediction. In: *IJCAI* (2005)
17. Slaney, M.: Web-scale multimedia analysis: does content matter? *IEEE Multimed.* **18**(2), 12–15 (2011)
18. Wang, D., Deng, S., Xu, G.: Sequence-based context-aware music recommendation. *Inf. Retr. J.* **21**(2–3), 230–252 (2018)
19. Wang, X., Wang, Y.: Improving content-based and hybrid music recommendation using deep learning. In: *ACM Multimedia*, pp. 627–636 (2014)
20. Zangerle, E., Gassler, W., Specht, G.: Exploiting twitter’s collective knowledge for music recommendations. In: *#MSM*, pp. 14–17 (2012)



# Online Collaborative Filtering with Implicit Feedback

Jianwen Yin<sup>1,2</sup>, Chenghao Liu<sup>3(✉)</sup>, Jundong Li<sup>4</sup>, BingTian Dai<sup>3</sup>,  
Yun-chen Chen<sup>3</sup>, Min Wu<sup>5</sup>, and Jianling Sun<sup>1,2(✉)</sup>

- <sup>1</sup> School of Computer Science and Technology, Zhejiang University, Hangzhou, China  
`{yinjw, sunjl}@zju.edu.cn`
- <sup>2</sup> Alibaba-Zhejiang University Joint Institute of Frontier Technologies,  
Hangzhou, China
- <sup>3</sup> School of Information Systems, Singapore Management University,  
Singapore, Singapore  
`{chliu, btdai, jeanchen}@smu.edu.sg`
- <sup>4</sup> Computer Science and Engineering, Arizona State University, Tempe, USA  
`jundong.li@asu.edu`
- <sup>5</sup> Data Analytics Department Institute for Infocomm Research,  
A-Star, Singapore, Singapore  
`wumin@i2r.a-star.edu.sg`

**Abstract.** Studying recommender systems with implicit feedback has become increasingly important. However, most existing works are designed in an offline setting while online recommendation is quite challenging due to the one-class nature of implicit feedback. In this paper, we propose an online collaborative filtering method for implicit feedback. We highlight three critical issues of existing works. First, when positive feedback arrives sequentially, if we treat all the other missing items for this given user as the negative samples, the mis-classified items will incur a large deviation since some items might appear as the positive feedback in the subsequent rounds. Second, the cost of missing a positive feedback should be much higher than that of having a false-positive. Third, the existing works usually assume that a fixed model is given prior to the learning task, which could result in poor performance if the chosen model is inappropriate. To address these issues, we propose a unified framework for Online Collaborative Filtering with Implicit Feedback (OCFIF). Motivated by the regret aversion, we propose a divestiture loss to heal the bias derived from the past mis-classified negative samples. Furthermore, we adopt cost-sensitive learning method to efficiently optimize the implicit MF model without imposing a heuristic weight restriction on missing data. By leveraging meta-learning, we dynamically explore a pool of multiple models to avoid the limitations of a single fixed model so as to remedy the drawback of manual/heuristic model selection. We also analyze the theoretical bounds of the proposed OCFIF method and conduct extensive experiments to evaluate its empirical performance on real-world datasets.

## 1 Introduction

Recommender systems aim to alleviate information overload by providing personalized suggestions from a superabundant of choices based on the historical behaviour. Among various recommendation algorithms, Collaborative Filtering (CF), an approach that uses known preferences of some users to make predictions to the unknown preferences of other users, has been widely used as one of the core learning techniques in building real-world recommender systems. The prevalent of E-commerce and social media sites generate massive data at an unprecedented rate. More than 10 million transactions are made per day in eBay<sup>1</sup> [1] and about half a billion tweets are generated every day [12]. Such data is temporally ordered, high-velocity and time varying. Unfortunately, traditional CF based methods adopt batch machine learning techniques which assume all training data are provided prior to model training. Such assumption makes them unsuitable and non-scalable for real-world applications for the following reasons. First, the user-item interactions usually arrive sequentially and periodically while batch learning model has to be retrained from scratch whenever new samples are received, making the training process extremely expensive. Second, whenever a new user/item is added to the system, batch learning cannot handle such changes immediately without involving an expensive re-training process. Third, it is common that user preferences are likely to change through time, but it is difficult for a batch learning model to capture the changes. Therefore, it is imperative to develop real-time scalable recommendation algorithms.

Recent years have witnessed some emerging studies for online recommendation methods [1, 10, 21]. These methods generally follow the paradigm of Matrix Factorization (MF) model which associates each user and item with a latent vector respectively and assume that the corresponding rating is estimated by the vector inner product. These works formulate the recommendation task as a rating prediction problem which is denoted by explicit feedback. Nevertheless, implicit feedback, such as monitoring clicks, view times, purchases, etc, is much cheaper to obtain than explicit feedback, since it comes with no extra cost for the user and thus is available on a much larger scale. Compared to explicit ratings, implicit feedback is much more challenging due to the natural scarcity of negative feedback (also known as the one-class problem). One popular solution to solve this problem is to select some negative instances from unlabeled entries [2, 14]. However, this adversely decreases the efficacy of the predictive model due to insufficient data coverage. Another solution [8] is to contrast the positive feedback against all the non-observed interactions. However, this strategy significantly increases the computation cost. A state-of-the-art MF method for implicit feedback is the eALS [7], which treats all missing data as the negative feedback but with a heuristic weight. Despite its success in dealing with batch learning setting, it is challenging to develop online recommendation methods with implicit feedback for the following reasons: (i) when positive feedback arrives sequentially, if we treat all the other missing items for this given user as

---

<sup>1</sup> <http://www.webretailer.com/articles/ebay-statistics.asp>.

the negative feedback, the mis-classified items will incur a large deviation since some items might appear as the positive feedback in the subsequent rounds; (ii) the cost of missing a positive target is much higher than that of having a false-positive [23]; (iii) the existing works usually assume that prior to the learning task, a fixed model is given either by manual selection or via cross validation. This could result in poor performance if the chosen model is inappropriate in a new environment, which happens commonly for real-world applications since user preferences and item attributes dynamically change over time.

To address these issues, we propose a unified framework for Online Collaborative Filtering with Implicit Feedback (OCFIF). First, motivated by the regret aversion [16], we propose a divestiture loss to heal the bias derived from the past mis-classified negative samples. Next, we utilize cost-sensitive learning method [3] to efficiently optimize the implicit MF model without imposing a heuristic weight restriction on missing data. Finally, we leverage meta-learning method [18] to explore a pool of multiple models, which are assigned with weights according to their real-time performance, to remedy the drawback of using a single fixed model by existing methods that often suffer considerably when the single model is inappropriate. In this way, the selection of the optimal model is adaptive and evolving according to the streaming data. By leveraging divestiture loss, cost-sensitive learning and meta-learning, our implicit MF objective function integrates them into a joint formulation. We theoretically analyze the regret bounds of the proposed framework. To validate the efficacy of the proposed method, we conduct extensive experiments by evaluating the proposed algorithms on real-world datasets, showing that our method outperforms the existing state-of-the-art baselines.

## 2 Related Work

The proposed work in this paper is mainly related to following two directions: (i) recommendation with implicit feedback; (ii) online recommender systems.

### 2.1 Recommendation with Implicit Feedback

While early literature on recommendation has largely focused on explicit feedback [9, 15], recent attention is increasingly shifting towards implicit data [7, 8, 23]. We can categorize previous works for implicit feedback into two types: sample-based learning and whole-data based learning. The first type samples negative instances from missing data [2, 14]. The BPR method [14] randomly samples negative instances from missing data, optimizing the model with a pairwise ranking loss. Later on, [2] improves BPR with a better negative sampler by additionally leveraging view data in E-commerce. By reducing negative examples in training process, these sample-based methods are more efficient in training, but the convergence speed is slower. The second type treats all missing entries as negative instances [6–8]. For example, the WALS [8] method models all missing entries as negative instances, assigning them with a uniform lower weight in

point-wise regression learning. Recently, [6] develops efficient learning algorithms for any non-uniform weight on missing data. These whole-data based methods can achieve a higher data coverage, but the training cost is much more expensive. Although the aforementioned batch learning methods achieve relatively accurate prediction performance, they often suffer poor efficiency and scalability issues for the online recommendation tasks.

## 2.2 Online Recommender Systems

There are a variety of research works studying online recommendation algorithms for explicit feedback. Most of these works focus on how to update the model efficiently in online setting. Inspired by online multi-task learning, OMTCF algorithm [20] treats users as tasks and update the models of multiple users simultaneously. [9] and [11] algorithms consider second-order information to achieve faster convergent rate. Beyond the efficient updating problem, RKMF model [15] focuses on solving the new user/item problem in the stream setting. [1] employs a continuous markov process for each time-varying user/item latent vector to solve the user interest drift problem. Recently, [21] solves new user/item, user interest drift and overload problem in a single framework with probabilistic matrix factorization model. By contrast, there are few studies on implicit online recommendation. [19] develops a fast incremental Matrix Factorization algorithm for recommendation with positive-only feedback. However, modeling only the positive feedback results in biased representations in user profile [8]. To this end, [7] proposes an online implicit matrix factorization method, called eALS, which models all the missing data as negative feedback. Although eALS could update model in online setting, the basic model needs to be trained offline on a large amount of historical data first (compared to the amount of data for online update). Otherwise, the performance of eALS will significantly drop which limits model's flexibility and scalability for real applications.

## 3 Online Collaborative Filtering with Implicit Feedback

### 3.1 Problem Formulation

First, we motivate the problem by introducing the formulation of implicit MF model. For a user-item rating matrix  $\mathbf{A} \in \mathbb{R}^{m \times n}$ ,  $m$  and  $n$  denote the number of users and items respectively,  $\Omega$  denotes the set of user-item pairs that have interactions. In the implicit setting, we define the observation matrix  $\mathbf{R}$ , where  $R_{ij} = 1$  if  $(i, j) \in \Omega$ , and  $R_{ij} = 0$  otherwise. MF maps both users and items into a joint latent feature space of  $k$  dimension. Formally, let  $\mathbf{U} \in \mathbb{R}^{k \times m}$  be the latent factor corresponding to the users, where the  $i$ -th column  $\mathbf{u}_i \in \mathbb{R}^k$  is the latent factor for user  $i$ . Similarly, let  $\mathbf{V} \in \mathbb{R}^{k \times n}$  be the latent factor for the items, where the  $j$ -th column  $\mathbf{v}_j \in \mathbb{R}^k$  is the latent factor for item  $j$ . In this work, we cast implicit MF as an online learning problem. On each round  $t$ , an observed matrix entry  $r_{ij}^t$  is revealed, where  $(i, j) \in \Omega$ . The goal of OCFIF is to update  $\mathbf{u}_i^t$

and  $\mathbf{v}_j^t$  such that  $r_{ij}^t \approx (\mathbf{u}_i^t)^\top \mathbf{v}_j^t$ . The existing online recommendation methods [9, 11] then alternatively update  $\mathbf{u}_i^t$  and  $\mathbf{v}_j^t$  while keeping the other one fixed by minimizing the incurred loss  $\ell(r_{ij}^t, \hat{r}_{ij}^t)$ , where  $\hat{r}_{ij}^t = (\mathbf{u}_i^t)^\top \mathbf{v}_j^t$ .

However, this learning process is not suitable for online recommendation with implicit feedback for the following reasons: (i) when positive feedback arrives sequentially, if we treat all the other missing items for this given user as the negative feedback, it will significantly increase the time for updating model which hinders it from deploying online. Moreover, the data we treat as negative feedback can appear as positive feedback later, which brings a non-negligible side-effect to the model; (ii) the cost of missing a positive target is much higher than that of having a false-positive; (iii) the existing works usually assume that prior to the learning task, a fixed model is given either by manual selection or via cross validation. This could result in poor performance if the chosen model is inappropriate in a new environment, which is widely observed in real-world applications since user preferences and item attributes dynamically change.

### 3.2 OCFIF Framework

Due to the one-class problem, the model cannot receive any negative samples for training in online setting. Thus, we follow the conventional assumption that treats all the other missing items for the given user at each round as the negative feedback. To address the above issues, we propose a unified framework for Online Collaborative Filtering with Implicit Feedback (OCFIF) as shown in Fig. 1. First, motivated by the regret aversion [16], we propose a divestiture loss to heal the bias derived from the past mis-classified negative samples. Next, we develop a cost-sensitive learning method [3] that efficiently optimizes the implicit MF model without imposing a heuristic weight restriction on missing data. Finally, we utilize meta-learning [18] to explore a pool of multiple models, which are assigned with weights according to their real-time performance, to remedy the drawback of using a single fixed model by existing methods as their performance often degrade considerably when the single model is inappropriate. In this way, the selection of the optimal model is adaptive and evolving according to the streaming data. By leveraging divestiture loss, cost-sensitive learning and meta-learning, the proposed framework integrates them into a joint formulation.

**Divestiture Loss.** To tackle the issue of mis-classified negative samples, we adopt the regret aversion [16] idea to amend the bias. Regret aversion is originated from decision theory, which encourages to anticipate regret when facing a decision and thus incorporate their desire to eliminate or reduce this possibility in their choice. It has been found to influence choices in a variety of important domains including health-related decisions, consumer behavior, and investment decisions. In our setting, we hypothesize that model should attach a higher weight to the positive samples that were mis-classified as the negative samples in the past than the other data. Therefore, we explicitly deal with this cost as the divestiture loss and integrate it into the optimization problem. Formally, we

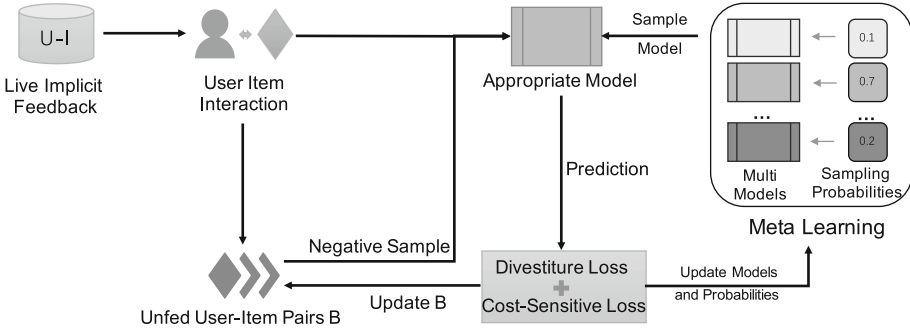


Fig. 1. The OCFIF framework.

denote the set of historical user-item pairs which are treated as negative samples before round  $t$  as  $N_t$ , then the divestiture loss is formulated as:

$$\xi(r_{ij}^t, \hat{r}_{ij}^t) = (1 + \lambda \mathbb{I}[(i, j) \in N_t]) \ell(r_{ij}^t, \hat{r}_{ij}^t), \tag{1}$$

where  $\ell$  could be any convex loss function and we instantiate it as the  $\epsilon$ -insensitive loss  $\ell(r_{ij}^t, \hat{r}_{ij}^t) = \max(|r_{ij}^t - \hat{r}_{ij}^t| - \epsilon, 0)$ .  $\mathbb{I}[\cdot]$  is the indicator function that equals to 1 if the statement holds; and 0 otherwise.  $\lambda \geq 0$  is a hyper-parameter that balance the original loss and the extra penalty. If  $\lambda = 0$ , the extra penalty disappears and the loss function becomes the conventional one.  $\mathbb{I}[(i, j) \in N_t]$  indicates whether the user-item pair  $(i, j)$  has been mis-classified as the negative sample in the past.

In order to meet high efficiency requirement of online recommendation and reduce the number of mis-classified samples, we denote the set of user-item pairs which have not been fed to the model before round  $t$  as  $B_t$ , then sample  $Z$  negative instances from  $B_t$  for model update. A naive sample strategy is to uniformly sample from  $B_t$  which assumes each missing entry is negative feedback with equal probability. We follow the assumption in [7, 13] that popular items are more likely to be known by users in general, and thus a miss on a popular item is more likely to be truly negative. In this way, we sample negative instances according to the global item popularity. Let  $D_t$  denotes the set of user-item pairs that revealed before round  $t$ . The sampling distribution of item  $j$  is defined as:

$$p(j) = \frac{\sum_{i'=1}^m \mathbb{I}[(i', j) \in D_t]}{\sum_{j'=1}^n \sum_{i'=1}^m \mathbb{I}[(i', j') \in D_t]}. \tag{2}$$

**Cost-Sensitive Learning.** Although  $\xi$  can deal with the issue of mis-classified negative samples, it equally penalizes the mistakes on both positive and negative entries. However, in the implicit feedback scenario, the cost of missing a positive target is much higher than that of having a false-positive. Thus, we assume  $r_{ij} = \mathbb{I}[\hat{r}_{ij} \geq q]$ , where  $q \in [0, 1]$  is a threshold, and adopt cost-sensitive learning method with a more appropriate metric, such as the *sum* of weighted *recall* and *specificity*.

$$sum = \mu_p \times recall + \mu_n \times specificity, \tag{3}$$

where  $\mu_p + \mu_n = 1$  and  $0 \leq \mu_p, \mu_n \leq 1$ . In general, the higher the *sum*, the better the performance. Besides, another appropriate metric is to measure the total cost of algorithm:

$$cost = c_p \times M_p + c_n \times M_n, \tag{4}$$

where  $M_p$  denotes the number of false negatives and  $M_n$  denotes the number of false positives.  $c_p + c_n = 1$  and  $0 \leq c_p, c_n \leq 1$  are the misclassification cost of positive and negative, respectively. In general, the lower the *cost* value, the better the performance.

**Lemma 1.** *The goal of maximizing the weighted sum in (3) or minimizing the weighted cost in (4) is equivalent to minimizing the following objective:*

$$\sum_{r_{ij}=+1} \rho \mathbb{I}(\hat{r}_{ij} \leq q) + \sum_{r_{ij}=0} \mathbb{I}(\hat{r}_{ij} > q), \tag{5}$$

where  $\rho = \frac{\eta_p T_n}{\eta_n T_p}$  for the maximization of the weighted sum, and  $\rho = \frac{c_p}{c_n}$  for the minimization of the weighted cost.  $T_p$  and  $T_n$  are the number of positive examples and negative examples, respectively.

Lemma 1 gives the explicit objective function to optimize, but the indicator function is not convex. To solve this problem, we replace this objective function by its convex surrogate and derive the following two cost-sensitive loss functions:

$$\begin{aligned} \ell^I(r_{ij}, \hat{r}_{ij}) &= \rho \mathbb{I}(r_{ij}=1) \ell(\hat{r}_{ij}, 1) + \mathbb{I}(r_{ij}=0) \ell(\hat{r}_{ij}, 0), \\ \ell^{II}(r_{ij}, \hat{r}_{ij}) &= \mathbb{I}(r_{ij}=1) \ell(\hat{r}_{ij}, \rho) + \mathbb{I}(r_{ij}=0) \ell(\hat{r}_{ij}, 0). \end{aligned}$$

We could find that the slope of  $\ell^I(r_{ij}, \hat{r}_{ij})$  changes for specific class, leading to more “aggressive” updating while the required margin of  $\ell^I(r_{ij}, \hat{r}_{ij})$  changes for specific class, resulting in more “frequent” updating.

**Meta-learning.** By introducing cost-sensitive loss and divestiture loss, we can solve the problem of implicit feedback in online recommendation. However, how to decide the value of hyper-parameter like  $\rho$  remains an issue. Typically, in a batch learning setting, one can choose hyper-parameters with manual selection or via cross validation prior to the learning task, which is impossible for online learning setting. Moreover, in the real-world online recommender systems, user preferences and item attributes dynamically change. To address this issue, we adopt the meta-learning method [22] to exploit the benefit of multiple implicit MF models. The motivation is that if multiple implicit MF models with a number of hyper-parameters are learned simultaneously, there must exist one setting that is most appropriate to the streaming data. Specifically, take the hyper-parameter  $\rho$  as an example, we construct a pool of multiple values of parameter  $\rho$  by discretizing  $(0, 1)$  into  $S$  evenly distributed values  $\frac{1}{S+1}, \dots, \frac{s}{S+1}, \dots, \frac{S}{S+1}$  and setting  $\rho_s$  to  $(1 - \frac{s}{S+1}) / (\frac{s}{S+1})$ .



A remaining issue is how to choose appropriate implicit MF model from  $S$  candidates for prediction and update them at each round. We apply the Hedge algorithm [4] and randomly select a model according to a distribution  $\mathbf{p}_t = (p_t^1, \dots, p_t^S)$  such that  $\sum_s p_t^s = 1$  and  $p_t^s \geq 0$ . The sampling probabilities represents the online predictive performance of each model which is defined as

$$p_t^s = \frac{\exp(\gamma M_t^s)}{\sum_{s=1}^S \exp(\gamma M_t^s)}, \quad s = 1, \dots, S, \tag{6}$$

where  $\gamma > 0$  is a temperature hyper-parameter, and  $M_t^s$  is an online performance measure on historical data. Here we choose two commonly used performance measure in recommendation with implicit feedback task: F-measure and AUC [14, 23]. However, both of these two performance measures are non-decomposable which makes it significantly challenging to directly optimize them in the online process. Motivated by [22], we propose the following update methods.

**Update F-measure.** For each entry  $r_{ij}^t$  at round  $t$ , the model produces an  $N$ -size ranking list of items for user  $i$ . We denote  $h_t$  as the hit result that equals to 1 if item  $j$  appears in the ranking list and 0 otherwise. Then the F-measure can be computed by  $F@N_{t+1} = \frac{2 \sum_{\tau=1}^t r_{ij}^\tau h_\tau}{\sum_{\tau=1}^t r_{ij}^\tau + \sum_{\tau=1}^t h_\tau}$ . However, directly calculating the online F-measure by going through all entries in history is expensive. Therefore, we introduce an incremental calculation method according to [22]. Let  $a_t = \sum_{\tau=1}^t r_{ij}^\tau h_\tau$  and  $c_t = \sum_{\tau=1}^t (r_{ij}^\tau + h_\tau)$ , then we can calculate  $F@N_{t+1} = \frac{2a_t}{c_t}$  and update  $a_t$  and  $c_t$  incrementally by

$$a_{t+1} = \begin{cases} a_t + 1, & \text{if } r_{ij}^{t+1} = 1 \text{ and } h_{t+1} = 1, \\ a_t, & \text{otherwise;} \end{cases}$$

$$c_{t+1} = \begin{cases} c_t + 2, & \text{if } r_{ij}^{t+1} = 1 \text{ and } h_{t+1} = 1, \\ c_t + 1, & \text{if } r_{ij}^{t+1} = 1 \text{ or } h_{t+1} = 1, \\ c_t, & \text{otherwise.} \end{cases}$$

**Update AUC.** Similar to F-measure case, directly calculating the online AUC is difficult, which requires to compare the present entry to historically received entries. To avoid storing the prediction results of all entries, we introduce two  $E$ -length hash tables  $L_+^t$  and  $L_-^t$  with ranges  $(0, 1/E), (1/E, 2/E), \dots, ((E-1)/E, 1)$ . For  $e \in 1, \dots, E$ ,  $L_+^t[e]$  and  $L_-^t[e]$  store the number of positive entries and negative entries before round  $t$  respectively, whose prediction result  $\hat{r}_{ij}$  are such that  $1/(1 + \exp(-\hat{r}_{ij})) \in [(e-1)/E, e/E)$ . Then, we can approximately update the online AUC using the two hash tables according to [22]. In particular, if  $r_{ij}^{t+1} = 1$ ,  $AUC_{t+1} = \frac{N_+^t}{N_+^t + 1} AUC_t + \frac{1}{(N_+^t + 1)N_-^t} \left( \sum_{k=1}^e L_-^t[k] + \frac{L_-^t[e+1]}{2} \right)$ , where  $e$  is the largest index such that  $e/E \leq 1/(1 + \exp(-\hat{r}_{ij}^{t+1}))$ ; if  $r_{ij}^{t+1} = 0$ , we have  $AUC_{t+1} = \frac{N_-^t}{N_-^t + 1} AUC_t + \frac{1}{N_+^t(N_-^t + 1)} \left( \sum_{k=e+1}^{E-1} L_-^t[k] + \frac{L_+^t[e]}{2} \right)$ , where  $e$  is the smallest index such that  $e/E \geq 1/(1 + \exp(-\hat{r}_{ij}^{t+1}))$ .

Now we can summarize our proposed framework in Algorithm 1. We could use any online optimization methods [17] to solve the objective in (1). A default choice is online gradient descent (OGD) [17] due to its simplicity and popularity. At each round  $t$ , we alternatively update  $\mathbf{u}_i^t$  and  $\mathbf{v}_j^t$  while keeping the other matrix fixed, the update rules are:

$$\mathbf{u}_i^{t+1} = \mathbf{u}_i^t - \eta_t \nabla_{\mathbf{u}_i} \xi(r_{ij}^t, \hat{r}_{ij}^t), \tag{7}$$

$$\mathbf{v}_j^{t+1} = \mathbf{v}_j^t - \eta_t \nabla_{\mathbf{v}_j} \xi(r_{ij}^t, \hat{r}_{ij}^t), \tag{8}$$

where  $\eta_t$  is the learning rate.

---

**Algorithm 1.** The OCFIF Framework

---

**Input:** the number of models  $S$   
 Randomly initialize  $\mathbf{U}_s, \mathbf{V}_s$  for  $s = 1, 2, \dots, S$ ,  $\mathbf{p}_1 = (1/S, 1/S, \dots, 1/S)$ ;  
**for**  $t = 1, 2, \dots, T$  **do**  
   Receive an observed entry  $r_{i_t j_t}$ ;  
   Sample negative item set  $Z$  from  $B_t \setminus \{(i_t, j_t)\}$ ;  
   **for all**  $r_{ij} \in \{\{r_{i_t j_t}\} \cup \{r_{i' j'} = 0 | i' = i, j' \in Z\}\}$  **do**  
     Sampling a model  $s$  according to the distribution in (6);  
     Compute prediction  $\hat{r}_{ij}$  and loss  $\xi$ ;  
     **for**  $s = 1, 2, \dots, S$  **do**  
       Update  $\mathbf{u}_{s,i}, \mathbf{v}_{s,j}$  with Eqs. (7), (8);  
       Update the performance  $M_s$ ;  
     **end for**  
     Update  $\mathbf{p}_{t+1}$  with Eq. (6);  
   **end for**  
   Update  $B_{t+1} \leftarrow B_t \setminus \{(i_t, j_t) \cup Z\}$ ,  $N_{t+1} \leftarrow N_t \cup Z$  and  $D_{t+1} \leftarrow D_t \cup (i_t, j_t)$ ;  
**end for**

---

### 3.3 Theoretical Analysis

We now analyze the theoretical performance of the OCFIF framework in terms of online regret bound analysis. To ease our discussion, we simplify some notations in our analysis as in Table 1.

We denote by  $\mathcal{S}$  the set of indexes that correspond to the trials when a loss happens,  $\mathcal{S} = \{t | \xi_t(\mathbf{w}_t) > 0\}$ . Similarly, we denote by  $\mathcal{S}_p = \{t | \xi_t(\mathbf{w}_t) > 0 \text{ and } y_t = 1\}$ ,  $\mathcal{S}_n = \{t | \ell_t(\mathbf{w}_t) > 0 \text{ and } y_t = 0\}$ ,  $S_p = |\mathcal{S}_p|$ ,  $S_n = |\mathcal{S}_n|$ .

**Table 1.** Simplification of notations.

Notations		Meaning
$\mathbf{v}_j^t$ or $\mathbf{u}_i^t$	$\mathbf{x}_t$	input
$\mathbf{u}_i^t$ or $\mathbf{v}_j^t$	$\mathbf{w}_t$	current status
$\mathbf{u}_i^{t+1}$ or $\mathbf{v}_j^{t+1}$	$\mathbf{w}_{t+1}$	solution
$\mathbf{u}$ or $\mathbf{v}$	$\mathbf{w}$	variable
$r_{ij}^t$	$y_t$	target

**Theorem 1.** Let  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_T, y_T)$  be a sequence of input-target pairs, where  $\mathbf{x}_t \in \mathbb{R}^k$  and  $G = \max_t \|\mathbf{x}_t\|^2$ ,  $y_t \in \{0, 1\}$ . Let  $\mathbf{w}_1, \dots, \mathbf{w}_T$  be a sequence of vectors obtained by the proposed algorithm. Then for any  $\mathbf{w} \in \mathbb{R}^k$ , by setting  $\eta = \frac{\|\mathbf{w}\|}{\sqrt{G[(\rho+\rho\lambda)^2 S_p + S_n]}}$  for  $\xi^I$ ,  $\eta = \frac{\|\mathbf{w}\|}{\sqrt{G[(1+\lambda)^2 S_p + S_n]}}$  for  $\xi^{II}$ , we then have the bounds of the proposed algorithms:

$$\begin{aligned} \sum_{t=1}^T \xi_t^I(\mathbf{w}_t) - \sum_{t=1}^T \xi_t^I(\mathbf{w}) &\leq \|\mathbf{w}\| \sqrt{G[(\rho + \rho\lambda)^2 S_p + S_n]}, \\ \sum_{t=1}^T \xi_t^{II}(\mathbf{w}_t) - \sum_{t=1}^T \xi_t^{II}(\mathbf{w}) &\leq \|\mathbf{w}\| \sqrt{G[(1 + \lambda)^2 S_p + S_n]}. \end{aligned}$$

*Proof.* Relying on the definition of OGD, we have

$$\begin{aligned} \|\mathbf{w}_{t+1} - \mathbf{w}\| &= \|\mathbf{w}_t - \eta \nabla \xi_t(\mathbf{w}_t) - \mathbf{w}\|^2 \\ &= \|\mathbf{w}_t - \mathbf{w}\|^2 + \eta^2 \|\nabla \xi_t(\mathbf{w}_t)\|^2 - 2\eta \nabla \xi_t(\mathbf{w}_t)(\mathbf{w}_t - \mathbf{w}). \end{aligned}$$

For the convexity of the loss function:  $\xi_t(\mathbf{w}_t) - \xi_t(\mathbf{w}) \leq \nabla \xi_t(\mathbf{w}_t)(\mathbf{w}_t - \mathbf{w})$ , we have the following:

$$\xi_t(\mathbf{w}_t) - \xi_t(\mathbf{w}) \leq \frac{\|\mathbf{w}_t - \mathbf{w}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}\|^2}{2\eta} + \frac{\eta}{2} \|\nabla \xi_t(\mathbf{w}_t)\|^2.$$

Summing over  $t = 1, \dots, T$ , gives

$$\sum_{t=1}^T (\xi_t(\mathbf{w}_t) - \xi_t(\mathbf{w})) \leq \frac{\|\mathbf{w}\|^2}{2\eta} + \frac{\eta}{2} \sum_{t=1}^T \|\nabla \xi_t(\mathbf{w}_t)\|^2.$$

When we adopt  $\xi^I$ , it is easy to see that  $\|\nabla \xi_t(\mathbf{w}_t)\| \leq \sqrt{G}$  if  $t \in \mathcal{S}_n$ ,  $\|\nabla \xi_t(\mathbf{w}_t)\| \leq \rho(1 + \lambda)\sqrt{G}$  if  $t \in \mathcal{S}_p$  and  $\|\nabla \xi_t(\mathbf{w}_t)\| = 0$  otherwise. Thus, we can obtain the bound  $\frac{\|\mathbf{w}\|^2}{2\eta} + \frac{\eta G[(\rho+\rho\lambda)^2 S_p + S_n]}{2}$ , by setting  $\eta = \frac{\|\mathbf{w}\|}{\sqrt{G[(\rho+\rho\lambda)^2 S_p + S_n]}}$ . Similarly, when we adopt  $\xi^{II}$ ,  $\|\nabla \xi_t(\mathbf{w}_t)\| \leq \sqrt{G}$  if  $t \in \mathcal{S}_n$ ,  $\|\nabla \xi_t(\mathbf{w}_t)\| \leq (1 + \lambda)\sqrt{G}$  if  $t \in \mathcal{S}_p$  and  $\|\nabla \xi_t(\mathbf{w}_t)\| = 0$  otherwise. Thus, we can obtain the bound  $\frac{\|\mathbf{w}\|^2}{2\eta} + \frac{\eta G[(1+\lambda)^2 S_p + S_n]}{2}$  by setting  $\eta = \frac{\|\mathbf{w}\|}{\sqrt{G[(1+\lambda)^2 S_p + S_n]}}$ .

Because  $S_p + S_n \leq T$ , we get a regret bound  $\sqrt{T}$ . From this theorem, our framework is guaranteed to converge to obtain the optimal average loss with respect to the online learning setting with divestiture loss and cost-sensitive loss. According to the theory of the Hedge algorithm [4], we can show that the OCFIF framework can achieve an optimal upper bound of regret by  $\sqrt{T \ln S/2}$  with  $S$  models after  $T$  iterations. This implies that it can asymptotically approach the most appropriate parameter setting and ensure the per-round regret vanishes over time in a sub-linear rate. The details of the proof can be found in [4].

## 4 Experiments

In this section, we conduct experiments with the aim of answering the following research questions:

**RQ1:** Does our proposed OCFIF framework outperform the state-of-the-art online implicit collaborative filtering methods?

**RQ2:** How do our sampling strategies perform? Which negative sampling strategy is better?

**RQ3:** How sensitive is our framework to hyper parameters?

In what follows, we first present the experimental settings, followed by answering the above three research questions.

### 4.1 Experimental Setting

**Dataset.** We experimented with three publicly accessible datasets: MovieLens<sup>2</sup>, Yelp<sup>3</sup> and Pinterest<sup>4</sup>. The characteristics of the three datasets are summarized in Table 2.

- **MovieLens.** This movie rating dataset has been widely used in recommendation task. We used the version containing one million ratings, where each user has at least 20 ratings. We transformed it into implicit data, where each entry is marked as 0 or 1 indicating whether the user has rated the item.
- **Yelp.** This is the Yelp Challenge data of user ratings on businesses. We use the filtered subset created by [7].
- **Pinterest.** This implicit feedback data is constructed by [5] for evaluating content-based image recommendation. The original data is very large but highly sparse. We filtered the dataset in the same way as for the MovieLens data that retained only users with at least 20 interactions (pins). This results in a subset of the data that contains 55,187 users and 1,500,809 interactions.

**Evaluation Metrics.** For experimental setup, each dataset is randomly divided into two parts: 80% for training and 20% for test. We repeat such a random permutation 10 times for each dataset and compute the average results of each algorithm over the 10 runs. For the metrics, The accuracy of a recommendation model is measured by two widely-used metrics, namely AUC and F-measure@N. AUC measures whether the items which are observed were held out during learning are ranked higher than unobserved items. F-measure@N is the weighted harmonic mean of precision and recall. In our experiments, we set  $N = 20$ . Basically, the higher these measures, the better the performance. For each metric, We report the score averaged by all the users.

<sup>2</sup> <http://grouplens.org/datasets/movielens/>.

<sup>3</sup> <https://www.yelp.com/dataset/challenge>.

<sup>4</sup> <https://sites.google.com/site/xueatalphabeta/academic-projects>.

**Baselines.** We compare three variants of the proposed OCFIF framework with the state-of-the-art algorithms for online recommendation tasks with implicit feedback as follows:

- **OBPR.** BPR [14] is a sample-based method that optimizes the pair-wise ranking between the positive and negative samples via SGD. We propose an Online BPR by online incremental learning [15]. We use a fixed learning rate, varying it and reporting the best performance.
- **ISGD** [19]. This is an incremental matrix factorization method for positive-only feedback. It learns by incremental SGD, which is acceptable for streaming data.
- **NN-APA** [9]. A second order online collaborative filtering algorithm.
- **OCFIF.** The proposed OCFIF framework. To examine the effectiveness of the framework on different components, we run two variants of OCFIF, OCFIF-C and OCFIF-CD. OCFIF-C only adopts cost-sensitive loss, and OCFIF-CD consider both cost-sensitive loss and divestiture loss.

For parameter settings, we adopt the same parameter tuning schemes for all the compared algorithm to enable fair comparisons. We perform grid search to choose the best parameters for each algorithm on the training set. For MF methods, the number of latent factors is tuned from  $\{10, 15, \dots, 50\}$ . For OCFIF, we search the ranges of values for negative sample size from  $\{1, 5, 10, \dots, 50\}$ , cost sensitive parameter  $\rho = \frac{c_p}{c_n}$  tuned with  $c_p$  from  $\{0.5, 0.55, \dots, 0.95\}$  with a stepsize of 0.5, and extra loss parameter  $\lambda \in \{0.1, 0.2, 0.5, 1, 2\}$ . We set the number of models  $S = 10$  and adopts uniform sampling strategy.

**Table 2.** Statistics of the evaluation datasets.

Dataset	#Interaction	#Item	#User
MovieLens	1000209	3706	6040
Yelp	731671	25815	25677
Pinterest	1500809	9916	55187

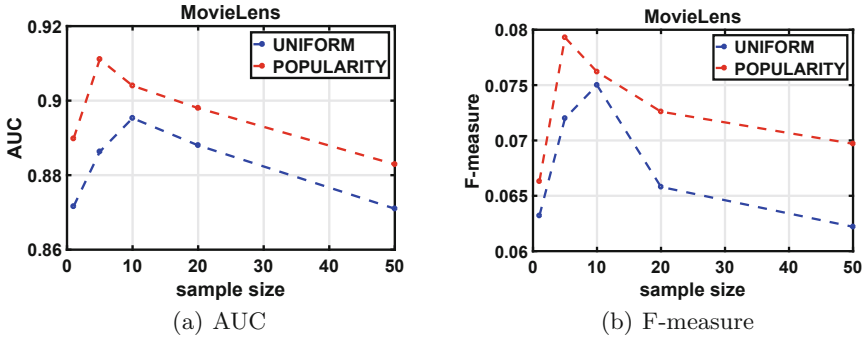
## 4.2 Performance Comparison (RQ1)

Table 3 summarizes the comparison results in terms of AUC and F-measure, from which we can draw several observations. First, it is clear to see that the proposed OCFIF framework and its variants significantly outperform OBPR, ISGD and NN-APA on all the datasets. This encouraging results validate the effectiveness of utilizing cost-sensitive loss. Furthermore, by examining the two variants of the proposed framework, we found that the OCFIF-CD with both divestiture loss and cost-sensitive loss significantly outperforms the OCFIF-C with only cost-sensitive loss. The reason is that divestiture loss could heal the bias derived from the past mis-classified negative samples. By further comparing OCFIF with its variants, we found that OCFIF is able to achieve the best performance.

This result highlights the importance of meta learning which dynamically explores a pool of multiple models to avoid the limitations of a single fixed model. Interestingly, we can observe that NN-APA outperform OBRP and ISGD in most cases, which is consistent with [9] that exploiting second-order information can improve the performance for explicit feedback. This could be a potential direction for OCFIF.

**Table 3.** Comparison of different algorithms in terms of AUC and F-measure for recommendation task.

Algorithm	Metrics	Movielens	Yelp	Pinterest
OBPR	AUC	0.8433 ± 0.0012	0.7754 ± 0.0004	0.6996 ± 0.0023
	F-measure	0.0487 ± 0.0004	0.0105 ± 0.0004	0.0210 ± 0.0003
ISGD	AUC	0.8578 ± 0.0009	0.8165 ± 0.0018	0.8620 ± 0.0018
	F-measure	0.0618 ± 0.0008	0.0150 ± 0.0003	0.0231 ± 0.0005
NN-APA	AUC	0.8600 ± 0.0003	0.8427 ± 0.0012	0.8810 ± 0.0008
OCFIF-C	F-measure	0.0581 ± 0.0008	0.0155 ± 0.0001	0.0246 ± 0.0004
	AUC	0.8953 ± 0.0010	0.8621 ± 0.0016	0.8901 ± 0.0022
OCFIF-CD	F-measure	0.0750 ± 0.0007	0.0162 ± 0.0003	0.0259 ± 0.0007
	AUC	0.9043 ± 0.0008	0.9105 ± 0.0011	0.9012 ± 0.0016
OCFIF	AUC	<b>0.9105 ± 0.0006</b>	<b>0.9126 ± 0.0012</b>	<b>0.9312 ± 0.0013</b>
	F-measure	<b>0.0809 ± 0.0004</b>	<b>0.0186 ± 0.0005</b>	<b>0.0275 ± 0.0002</b>



**Fig. 2.** Evaluation of different sample strategies: AUC (a) and F-measure (b).

### 4.3 Sampling Strategies Comparison (RQ2)

We compared OCFIF under different sampling strategies: uniform sampling and popularity based sampling. In particular, we set the sampling size gradually increasing from 0 to 50, and report the performance under different sampling

strategies. Figure 2(a) shows the performance evaluated by AUC and Fig. 2(b) shows the performance evaluated by F-measure. We can observe that popularity based sampling strategy is able to achieve the better performance than uniform sampling strategy. Moreover, the results show the same trend that better performance is obtained by increasing the sampling size. However, when sampling size is set too large, the performance suffers. The reason is that the likelihood of positive entries in negative sampling set will increase rapidly when sampling size is too large, which results in severely negative side-effect on the model.

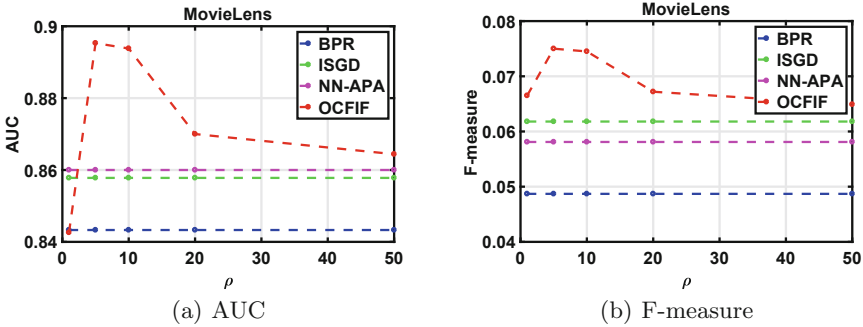


Fig. 3. Impact of cost sensitive parameter  $\rho$ .

#### 4.4 Evaluation of Parameter Sensitivity (RQ3)

For the proposed OCFIF framework, there are two key parameters: cost sensitive parameter  $\rho$  and divestiture loss parameter  $\lambda$ . Figures 3 and 4 show the results of parameter sensitive evaluations using different values of  $\rho$  and  $\lambda$ .

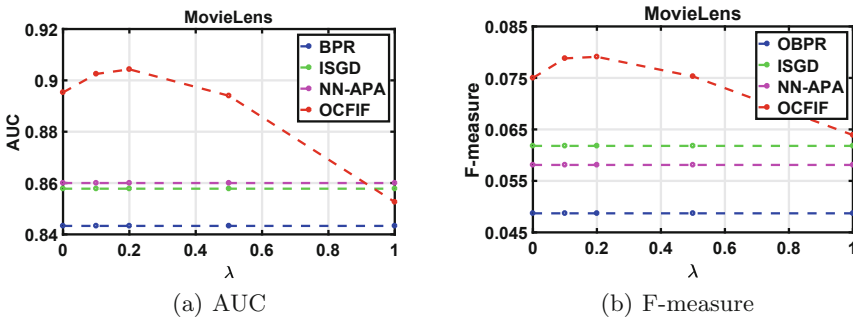


Fig. 4. Impact of divestiture loss parameter  $\lambda$ .

First of all, by examining the influence of cost sensitive parameter  $\rho$ , we found that the performance of framework is gradually improved with the increase of  $\rho$ .

This indicates the effectiveness of our cost sensitive learning. Moreover, when  $\rho$  is larger than 10, the performance starts to drop significantly. This reveals the drawback of over-updating of positive entries.

Second, by examining the influence of divestiture loss parameter  $\lambda$ , we found that the better performance is achieved by balancing between the impact of cost-sensitive loss and divestiture loss, while either a large or a small value of  $\lambda$  will adversely degrade the performance. This is primarily because that too large divestiture loss can cause excessive correction, then reducing the accuracy of model.

## 5 Conclusion

In this work, we propose a unified framework for online collaborative filtering with implicit feedback. Specifically, motivated by the regret aversion, we propose a divestiture loss to heal the bias derived from the past mis-classified negative samples. Furthermore, we adopt cost-sensitive learning method to efficiently optimize the implicit MF model without imposing a heuristic weight restriction on missing data. By leveraging meta-learning, we dynamically explore a pool of multiple models to avoid the limitations of a single fixed model so as to remedy the drawback of manual/heuristic model selection. We also analyze the theoretical bounds of the proposed OCFIF method, conduct extensive experiments and ablation studies, and achieve state-of-the-arts results on real-world datasets for online recommendation with implicit feedback task.

## References

1. Chang, S., et al.: Streaming recommender systems. In: Proceedings of the 26th International Conference on World Wide Web, pp. 381–389. International World Wide Web Conferences Steering Committee (2017)
2. Ding, J., Feng, F., He, X., Yu, G., Li, Y., Jin, D.: An improved sampler for Bayesian personalized ranking by leveraging view data. In: Companion of the Web Conference 2018 on the Web Conference 2018, pp. 13–14. International World Wide Web Conferences Steering Committee (2018)
3. Fernández, A., García, S., Galar, M., Prati, R.C., Krawczyk, B., Herrera, F.: Cost-sensitive learning. In: Learning from Imbalanced Data Sets, pp. 63–78. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-98074-4\\_4](https://doi.org/10.1007/978-3-319-98074-4_4)
4. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.* **55**(1), 119–139 (1997)
5. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web, pp. 173–182. International World Wide Web Conferences Steering Committee (2017)
6. He, X., Tang, J., Du, X., Hong, R., Ren, T., Chua, T.S.: Fast matrix factorization with non-uniform weights on missing data. arXiv preprint [arXiv:1811.04411](https://arxiv.org/abs/1811.04411) (2018)
7. He, X., Zhang, H., Kan, M.Y., Chua, T.S.: Fast matrix factorization for online recommendation with implicit feedback. In: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 549–558. ACM (2016)



8. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: Eighth IEEE International Conference on Data Mining, ICDM 2008, pp. 263–272. IEEE (2008)
9. Liu, C., Hoi, S.C., Zhao, P., Sun, J., Lim, E.P.: Online adaptive passive-aggressive methods for non-negative matrix factorization and its applications. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, pp. 1161–1170. ACM (2016)
10. Liu, C., Jin, T., Hoi, S.C., Zhao, P., Sun, J.: Collaborative topic regression for online recommender systems: an online and Bayesian approach. *Mach. Learn.* **106**(5), 651–670 (2017)
11. Lu, J., Hoi, S., Wang, J.: Second order online collaborative filtering. In: Asian Conference on Machine Learning, pp. 325–340 (2013)
12. Mathioudakis, M., Koudas, N.: Twittermonitor: trend detection over the twitter stream. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, pp. 1155–1158. ACM (2010)
13. Rendle, S., Freudenthaler, C.: Improving pairwise learning for item recommendation from implicit feedback. In: Proceedings of the 7th ACM International Conference on Web Search and Data Mining, pp. 273–282. ACM (2014)
14. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, pp. 452–461. AUAI Press (2009)
15. Rendle, S., Schmidt-Thieme, L.: Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In: Proceedings of the 2008 ACM Conference on Recommender Systems, pp. 251–258. ACM (2008)
16. Seiler, M., Seiler, V., Traub, S., Harrison, D.: Regret aversion and false reference points in residential real estate. *J. Real Estate Res.* **30**(4), 461–474 (2008)
17. Shalev-Shwartz, S., et al.: Online learning and online convex optimization. *Found. Trends® Mach. Learn.* **4**(2), 107–194 (2012)
18. Vanschoren, J.: Meta-learning: a survey. arXiv preprint [arXiv:1810.03548](https://arxiv.org/abs/1810.03548) (2018)
19. Vinagre, J., Jorge, A.M., Gama, J.: Fast incremental matrix factorization for recommendation with positive-only feedback. In: Dimitrova, V., Kuflik, T., Chin, D., Ricci, F., Dolog, P., Houben, G.-J. (eds.) UMAP 2014. LNCS, vol. 8538, pp. 459–470. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-08786-3\\_41](https://doi.org/10.1007/978-3-319-08786-3_41)
20. Wang, J., Hoi, S.C., Zhao, P., Liu, Z.Y.: Online multi-task collaborative filtering for on-the-fly recommender systems. In: Proceedings of the 7th ACM Conference on Recommender Systems, pp. 237–244. ACM (2013)
21. Wang, W., Yin, H., Huang, Z., Wang, Q., Du, X., Nguyen, Q.V.H.: Streaming ranking based recommender systems. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, pp. 525–534. ACM (2018)
22. Yan, Y., Yang, T., Yang, Y., Chen, J.: A framework of online learning with imbalanced streaming data (2017)
23. Yang, P., Zhao, P., Liu, Y., Gao, X.: Robust cost-sensitive learning for recommendation with implicit feedback. In: Proceedings of the 2018 SIAM International Conference on Data Mining, pp. 621–629. SIAM (2018)



# Subspace Ensemble-Based Neighbor User Searching for Neighborhood-Based Collaborative Filtering

Zepeng Li and Li Zhang<sup>(✉)</sup> 

School of Computer Science and Technology, Joint International Research Laboratory of Machine Learning and Neuromorphic Computing, Soochow University, Suzhou 215006, Jiangsu, China  
zpli520@stu.suda.edu.cn, zhanglim1@suda.edu.cn

**Abstract.** Neighborhood-based collaborative filtering (NCF) typically uses a similarity measure for finding similar users to a target user or similar products on which the target user rated. To find neighbor users, traditional similarity measures rely only on the ratings of co-rated items when calculating similarity of pairwise users. Some hybrid similarity measures can avoid this situation but they suffer from the time-consuming issue. To solve the mentioned issues, the current paper presents an effective method of subspace ensemble-based neighbor user searching (SENSUS) for NCF. First, three item subspaces are constructed, or interested, neither interested nor uninterested, and uninterested subspaces. In each subspace, we calculate the co-rating support values for pairwise users. Then, SENSUS combines three co-rating support values to get the total co-rating support values for pairwise users, which are utilized to generate direct neighbor users for a target user. For the target user, its neighbor users include direct and indirect ones in SENSUS, where its indirect neighbors are the direct neighbors of its direct neighbors. Experimental results on public datasets indicate that the proposed method is promising in recommender systems.

**Keywords:** Recommendation system · Collaborative filtering · Similarity measure · Neighbor user searching · Subspace ensemble

## 1 Introduction

Obviously, the Internet has become an important part of people's lives. The development of the Internet has made it possible to access different types of data online [14]. Therefore, people almost feel that they are surrounded by many

---

Supported in part by the National Natural Science Foundation of China under Grant No. 61373093, by the Soochow Scholar Project, by the Six Talent Peak Project of Jiangsu Province of China, and by the Collaborative Innovation Center of Novel Software Technology and Industrialization.

different items that are available for selection. However, each user can only visit a limited number of items since the number of items available is very large (such as the millions of items offered by Taobao). Moreover, it is very rare that different users would visit the same items. Recommendation system (RS) [9, 11] techniques have been successfully used to help people cope with the information overload problem, and established as an integral part of e-business domain over last decades [22]. The main task of RSs is to provide an individual user with personalized suggestions on products or items filtering through a large product or item space. Many RS algorithms have been developed in various applications, such as tourism [10], movies [8], music [19], and news [31].

Based on the modeling ways of RSs, the RS methods can be categorized into content-based and collaborative filtering algorithms. Content-based filtering algorithms try to analyze the profile of a target user, the profile of a target item and profiles of items that the target user preferred in past when recommending the target item to the target user. However, it is hard to analyze profiles in many applications such as multimedia data. Collaborative filtering (CF) algorithms are the most successful and widely used in RSs [1, 6, 28]. The assumption behind CF is that if some users have similar interesting items up to now, these users would have similar interests in future. CF is domain independent and more accurate than content-based filtering. There are two main approaches for recommending items in CF, called neighborhood-based CF (NCF) and model-based CF (MCF). NCF is simple, intuitive, and can provide an immediate response to a new user after receiving upon his/her feedback. In addition, NCF only works with a single parameter (the number of neighborhood) while MCF needs many parameters (say, learning parameter, regularization parameters, etc.).

Generally, NCF is to find a set of users, which are called neighbor users and similar to the given target user, through using a similarity measure. There are several well-known similarity measures including Pearson coefficient, cosine and their variants. Most of these measures consider ratings of the co-rated items between the target user and the other users. Therefore, it is insufficient to find the effective neighbor users especially for the sparse user who would rate only a small number of items. To remedy this, new hybrid similarity measures have been proposed, such as singularity-based similarity measure (SBSM) [7], proximity-impact-popularity (PIP) [3], new heuristic similarity model (NHSM) [21], CF based on the Bhattacharyya-coefficient (BCF) [24], and hybrid user similarity model (HUSM) [30]. Apparently, the computational complexity of hybrid similarity measures is much greater than that of single similarity measures. Thus, NCF with hybrid similarity measures is generally time-consuming.

To speed NCF, the idea of subspace has been adopted to find similar users [4, 29]. Generally, the original data would be partitioned to several subspaces, user subspaces, item subspaces or both. It has been shown that the clustering algorithms do a good job finding similar users for a target user [4, 5, 15, 17, 20, 29]. The users in the subspace to which the target user belongs are taken as its neighbor users. In doing so, we can reduce the computational complexity of NCF since the number of users in the subspace is significantly less

than the total number of users. In addition, it is shown that the RS based on ratings of similar users may be more accurate [13]. Moreover, item subspaces are also useful. Agarwal et al. found direct neighbor users in the subspace of items with “interested” rating values [2]. A users’ tree accessed on subspace (UTAOS) approach was presented to generate a neighbor user tree in the subspace of items with “interested” rating values [26]. Based on UTAOS, Hamidreza et al. developed a neighbor users by subspace clustering on collaborative filtering (NUSCCF) approach [16]. NUSCCF tries to construct three neighbor user trees in three subspaces and then generate direct and indirect neighbor users for the target user by searching these trees. However, since these methods select neighbor users too less to represent the true neighbor relationship of the target user. Moreover, these methods would occupy a lot of space because of the use of global tables.

To enhance the performance of NCF, this paper presents an effective method of subspace ensemble-based neighbor user searching (SENU) for NCF. Similar to NUSCCF, we first construct three item subspaces of interested, neither interested nor uninterested, and uninterested subspaces. In each subspace, we calculate the co-rating support values for pairwise users. SENUS generates direct neighbor users for each user according to the total co-rating support value combined from three subspaces. For a target user, its neighbor users include direct and indirect ones in SENUS, where its indirect neighbors are the direct neighbors of its direct neighbors.

The rest of this paper is organized as follows. Section 2 proposes the novel method, SENUS. Section 3 reports and analyzes experimental results. Finally, our conclusions are drawn in Sect. 4.

## 2 Subspace Ensemble-Based Neighbor User Searching

NCF, as we know, has two main stages: searching neighbor users, and predicting rating scores. Both stages require similarity measures and generally adopt the same similarity measure. To enhance the performance of NCF, this section presents the novel subspace ensemble-based neighbor user searching (SENU) method for the stage of neighbor searching. The framework of SENUS is shown in Fig. 1, where  $U = \{u_1, u_2, \dots, u_m\}$  and  $T = \{t_1, t_2, \dots, t_n\}$  are the sets of users and items, respectively,  $T_i$  is the subset of items rated by user  $u_i$ ,  $m$  is the number of users and  $n$  is the number of items.

First, SENUS generates three item subspaces according to interested (INT), NINU, and uninterested (UNI) items. Then, SENUS calculates co-rating support values for pairwise users in these three subspaces and combines the results. Finally, SENUS searches neighbor users according to the total co-rating supports.

### 2.1 Item Subspaces

To consider the user rating preference, SENUS covers the whole item space  $T$  using three item subspaces. Note that three item subspaces are not

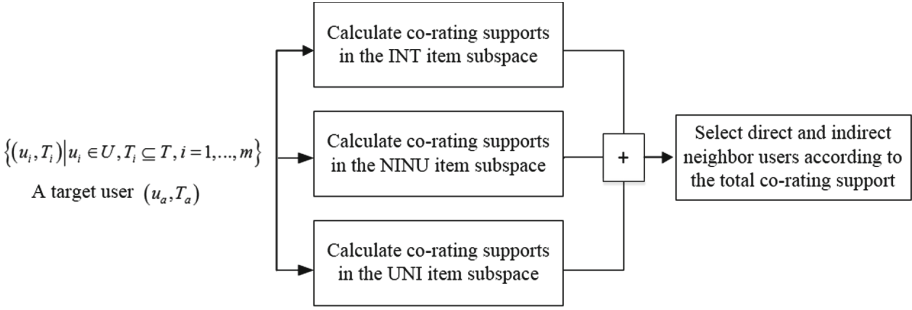


Fig. 1. Framework of SENSUS.

a partition of  $T$ . Let  $\mathbf{R} \in \mathbb{R}^{m \times n}$  be the user-item rating matrix, where the  $i$ -row and  $j$ -column element  $r_{ij} \in \{1, 2, \dots, r_{max}\}$  is the rating score made by user  $u_i$  on item  $t_j$ , and  $r_{max}$  is the highest rating.

Given a threshold  $\theta$ , we find a covering of  $T$  and form three subspaces,  $T^{INT}$ ,  $T^{NINU}$  and  $T^{UNI}$ , respectively. Without loss of generality, items with  $r_{ij} > \theta$  are considered as the INT ones, items with  $r_{ij} = \theta$  are thought as the NINU ones, and items with  $r_{ij} < \theta$  are the UNI ones. Namely,

$$T^{INT} = \{t_j | r_{ij} > \theta, j = 1, \dots, n\} \tag{1}$$

$$T^{NINU} = \{t_j | r_{ij} = \theta, j = 1, \dots, n\} \tag{2}$$

and

$$T^{UNI} = \{t_j | r_{ij} < \theta, j = 1, \dots, n\} \tag{3}$$

Empirically, we set  $\theta = \lceil r_{max}/2 \rceil$ , where the function  $\lceil \cdot \rceil$  rounds the value  $\cdot$  to the nearest integer towards infinity. Since the three item subspaces is a covering of  $T$ , we have

$$\begin{cases} T^{INT} \cap T^{NINU} \neq \emptyset \\ T^{INT} \cap T^{UNI} \neq \emptyset \\ T^{NINU} \cap T^{UNI} \neq \emptyset \\ T^{INT} \cup T^{NINU} \cup T^{UNI} = T \end{cases}$$

## 2.2 Co-rating Support

The Jaccard similarity is a common similarity measure for pairwise users [18], but the Jaccard similarity overly concerns the local information only on pairwise users themselves. Here, to measure the similarity of a pairwise user  $(u_i, u_p)$ , we modify the Jaccard similarity and define a term ‘‘co-rating support’’. In three item subspaces, we give the definition about this term.

**Definition 1.** Given the INT item set  $T^{INT}$ , a pairwise user  $(u_i, u_p)$ , and their rated interested item sets  $T_i^{INT} \subseteq T^{INT}$  and  $T_p^{INT} \subseteq T^{INT}$ , then the co-rating support of  $(u_i, u_p)$  in the INT item subspace is defined as:

$$s^{INT}(u_i, u_p) = \frac{|T_i^{INT} \cap T_p^{INT}|}{|T^{INT}|} \quad (4)$$

where  $0 \leq s^{INT}(u_i, u_p) \leq 1$ .

**Definition 2.** Given the NINU item set  $T^{NINU}$ , a pairwise user  $(u_i, u_p)$ , and their rated NINU item sets  $T_i^{NINU} \subseteq T^{NINU}$  and  $T_p^{NINU} \subseteq T^{NINU}$ , then the co-rating support of  $(u_i, u_p)$  in the NINU item subspace is defined as:

$$s^{NINU}(u_i, u_p) = \frac{|T_i^{NINU} \cap T_p^{NINU}|}{|T^{NINU}|} \quad (5)$$

where  $0 \leq s^{NINU}(u_i, u_p) \leq 1$ .

**Definition 3.** Given the UNI item set  $T^{UNI}$ , a pairwise user  $(u_i, u_p)$ , and their rated uninterested item sets  $T_i^{UNI} \subseteq T^{UNI}$  and  $T_p^{UNI} \subseteq T^{UNI}$ , then the co-rating support of  $(u_i, u_p)$  in the INT item subspace is defined as:

$$s^{UNI}(u_i, u_p) = \frac{|T_i^{UNI} \cap T_p^{UNI}|}{|T^{UNI}|} \quad (6)$$

where  $0 \leq s^{UNI}(u_i, u_p) \leq 1$ .

Here, the implied assumption is that the pairwise user  $(u_i, u_p)$  is similar if these two users have rated more INT items, NINU items, and UNI items. To unify the results of three subspaces, we define the total co-rating support of  $(u_i, u_p)$  as

$$s(u_i, u_p) = \frac{1}{3} (s^{INT}(u_i, u_p) + s^{NINU}(u_i, u_p) + s^{UNI}(u_i, u_p)) \quad (7)$$

where  $0 \leq s(u_i, u_p) < 1$ .

### 2.3 Neighbor User Searching

Now, our mission is to find neighbor users for a given target user  $u_a$ . We take the total co-rating support as the similarity measure and compute  $s(u_a, u_i)$ ,  $i = 1, \dots, m$  according to (7). It is easy to find the  $k$  direct neighbor users for  $u_a$  by sorting the total co-rating support with respect to  $u_a$ . Note that the greater  $s(u_a, u_i)$  is, the higher the similarity between  $u_a$  and  $u_i$ . Let  $N_a^D$  be the set of direct neighbor users for  $u_a$ .

To efficiently calculate similarity in the stage of prediction, we take into account indirect neighbor users except direct neighbors.

**Definition 4.** Given the set of users  $U = \{u_1, \dots, u_m\}$ , and their direct neighbor user sets  $N_i^D$ ,  $i = 1, \dots, m$ , the set of indirect neighbor users for  $u_a \in U$  is defined as:

$$N_a^I = \{u_i | u_i \in N_p^D \wedge u_p \in N_a^D \wedge u_i \notin N_a^D\} \quad (8)$$

Definition 4 states that an indirect neighbor user of  $u_a$  is the direct neighbor user of some direct neighbor user of  $u_a$ . Since the neighbor relation has transitivity, it is reasonable to choose these indirect neighbor users as the neighbors. Thus, the set of neighbor users for the target user  $u_a$  should be

$$N_a = N_a^D \cup N_a^I \quad (9)$$

## 2.4 Algorithm and Its Computational Complexity

SENSUS is the first stage in NCF for searching neighbor users in despite of which similarity measure is adopted in the prediction stage. The description of SENSUS is given in Algorithm 1. According to Algorithm 1, we can see that SENSUS consists of five steps. In Step 1, the computational complexity of generating three subspaces is  $O(3mn)$ , where  $m$  is the number of users and  $n$  is the number of items. In Step 2, the computational complexity of calculating the co-rating support in three subspaces is  $O(6n'm^2)$ , where  $n'$  is the maximum of item numbers rated by users. In Step 3, the computational complexity of selecting  $k$  direct neighbors is  $O(k \log k + m - k)$ . The computational complexity of generating indirect neighbor sets in Step 4 is  $O(k(k \log k + m - k))$ . Step 5 unifies direct and indirect neighbor users and has the computational complexity of  $O(k(k + 1))$ . Therefore, the total computational complexity of this algorithm is  $O(3mn + 6m^2n' + (k + 1)(k \log k + m))$ .

For a given target user  $u_a$ , SENSUS only provides its neighbor user set  $N_a$ . To recommend an unrated item  $t_j$  for  $u_a$ , we need to estimate the rating score of  $u_a$  on this item. The estimated rating score can be calculated as follow:

$$\hat{r}_{aj} = \bar{r}_a + \frac{\sum_{i \in N_a} \text{sim}(u_a, u_i)(r_{ij} - \bar{r}_i)}{\sum_{i \in N_a} |\text{sim}(u_a, u_i)|} \quad (10)$$

where  $\text{sim}(u_a, u_i)$  is a similarity measure, and  $\bar{r}_a$  is the average rating score of user  $u_a$ . If the estimated rating score is greater than or equal to the threshold  $\theta$ , we would recommend item  $t_j$  to user  $u_a$ .

---

### Algorithm 1: Subspace ensemble-based neighbor user searching

---

**Input:** User-item rating matrix  $\vec{R}$ , the user set  $U = \{u_i\}_{i=1}^m$ , the threshold  $\theta$ , the number of direct users  $k$ , and the target user  $u_a$ .

**Output:** The set  $N_a$  of neighbor users for the target user  $u_a$ .

1. Generate three item subspaces  $T^{INT}$  by (1),  $T^{INIU}$  by (2), and  $T^{UNI}$  by (3), respectively;
  2. Calculate the co-rating support values in three item subspaces according to Definitions 1, 2 and 3, respectively;
  3. Calculate the total co-rating support  $s(u_a, u_i), i = 1, \dots, m$  by (7) and generate the set  $N_a^D$  of direct neighbor users;
  4. For each  $u_g \in N_a^D$ , calculate the total co-rating support  $s(u_g, u_i), i = 1, \dots, m$  and generate the set  $N_a^I$  of indirect neighbor users by (8);
  5. Return  $N_a = N_a^D \cup N_a^I$ .
-

### 2.5 An Example

In the following, we give an example to illustrate the calculation of co-rating support and the result of finding the neighbor users for a target user.

Let  $U = \{u_1, u_2, \dots, u_7\}$  and  $T = \{t_1, t_2, \dots, t_8\}$ . Assume that the user-item rating matrix has the form in Fig. 2(a), where “-” means that users do not rate on items. Obviously, the highest score  $r_{max} = 5$  and the threshold  $\theta = 3$ . Items with ratings of 4 and 5 in  $\bar{R}$  are considered as the INT items, with rating of 3 as the NINU ones, and with ratings of 1 and 2 as the UNI ones. Then, the INT, NINU, and UNI item sets are  $T^{INT} = \{t_2, t_3, t_4, t_5, t_6, t_7, t_8\}$ ,  $T^{NINU} = \{t_1, t_2, t_4, t_6, t_7\}$ , and  $T^{UNI} = \{t_1, t_2, t_3, t_5, t_6\}$ , respectively.

Figures 2(b), (c) and (d) show the co-rating support matrices  $\bar{S}^{INT}$ ,  $\bar{S}^{NINU}$ , and  $\bar{S}^{UNI}$  of pairwise users  $(u_i, u_p), i, p = 1, \dots, 7$  in three subspaces, respectively, where “ $\infty$ ” denotes the co-rating support of pairwise users  $(u_i, u_i)$ . These matrices are symmetrical. Figure 2(e) gives the total co-rating support of all pairwise users. Based on Fig. 2(e), we can obtain the direct user sets for seven users. Let  $k = 2$ .

If user  $u_4$  is taken as the target user, the direct neighbor user set of  $u_4$  may be  $\{u_3, u_7\}$  or  $\{u_6, u_7\}$  by the observation on Fig. 2(e). If there are multiple candidate sets, we randomly select one from them. Without loss of generality, let  $N_4^D = \{u_6, u_7\}$ . Naturally, we can get the indirect neighbor users of  $u_4$  through merging the direct users of  $u_6$  and  $u_7$ . Finally, the neighbor user set of  $u_4$  may be  $N_4 = \{u_6, u_7\}$  with the probability of 1/6, and  $N_4 = \{u_3, u_6, u_7\}$  with the probability of 5/6. If we make another choice, or  $N_4^D = \{u_3, u_7\}$ , we would obtain  $N_4 = \{u_3, u_7\}$  with the probability of 1/6, and  $N_4 = \{u_3, u_6, u_7\}$  with the probability of 5/6. Therefore,  $N_4 = \{u_3, u_6, u_7\}$  has the highest probability being the neighbor user set of  $u_4$ .

Now, we carefully observe Fig. 2(a) and have a conclusion that users  $u_3, u_6$  and  $u_7$  are similar to  $u_4$  because they have the similar rating preference when rating items. Thus,  $u_3, u_6$  and  $u_7$  are effective neighbor users.

In SENUS, we calculate the similarity (total co-rating support) of pairwise users by combining the co-rating support of pairwise users in three item subspaces. Why do we consider the user rating preference? An intuitive idea is to compute the co-rating support of pairwise users in the whole item space instead of three subspaces, which has the form:

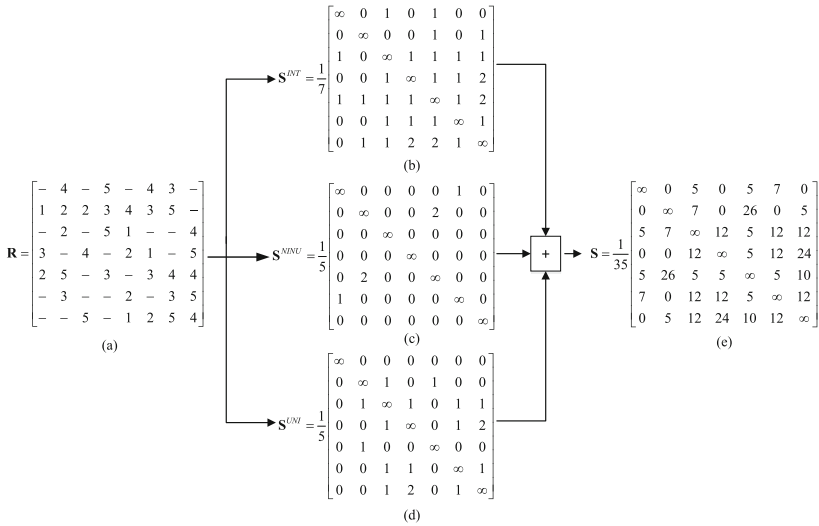
$$s^{Whole}(u_i, u_p) = \frac{|T_i \cap T_p|}{|T|} \tag{11}$$

where  $T_i$  is the rated item subset of  $u_i$ . Continue the example mentioned above. By using (11), the target user  $u_4$  would have two neighbor users  $u_2$  and  $u_7$ . Figure 2(a) shows that  $u_2$  and  $u_4$  have co-rated four items,  $t_1, t_3, t_5$  and  $t_6$ . However, their rating preference is totally different. In other words, the interests of  $u_4$  and  $u_2$  are opposite. Thus, we thought that  $u_2$  is not a good neighbor user of  $u_4$ . In a nutshell, SENUS can find effective neighbor users by taking into account the user rating preference.



### 3 Experiments

In order to validate the performance of SENUS, we perform comparison experiments on public datasets. All numerical experiments are performed on a personal computer with an Inter Core I5 processor with 8 GB RAM. This computer runs Windows 7, with Matlab R2012b.



**Fig. 2.** Neighbor users of Example, (a) user-item rating matrix  $\vec{R}$ , (b) co-rating support matrix  $\vec{S}^{INT}$  in the INT item subspace, (c) co-rating support matrix  $\vec{S}^{NINU}$  in the NINU item subspace, (d) co-rating support matrix  $\vec{S}^{UNI}$  in the UNI item subspace, and (e) total co-rating support matrix  $\vec{S}$ .

#### 3.1 Experimental Setting

Since SENUS is the method for searching neighbors, we need to combine it with other prediction schemes. Here, we design four SENUS-based methods, SENUS-AC-PCC, SENUS-NHSM, SENUS-BCF, and SENUS-HUSM by combining SENUS with AC-PCC [25], NHSM [21], BCF [23] and HUSM [30], respectively. In the following, we list all the compared methods.

**NHSM [21]:** NHSM is a hybrid similarity method, which distinguishes two users by utilizing the user rating preferences and considering co-rated items. In both searching and prediction stages, the hybrid similarity has been applied.

**AC-PCC [25]:** AC-PCC takes into account two additional weighting factors: the compromise factor and weighting factor, which is a hybrid similarity scheme. In both searching and prediction stages, AC-PCC uses the hybrid similarity.

**BCF [24]:** BCF is a hybrid similarity method based on the Bhattacharyya coefficient. BCF can utilize all ratings and provide a more reliable recommended list for active users. In the two stages of NCF, BCF uses the same similarity.

**HUSM [30]:** NUSM is a hybrid similarity measure, which considers the influence of all possible rated items, the non-linear relationship between variables, the asymmetry between users, and the rating preference of users. In the two stages of NCF, NUSM adopts the same similarity.

**UTAOS [26]:** UTAOS is a fast method for searching neighbor users, which constructs a neighbor user tree in the INT item subspace. In the prediction stage, UTAOS uses a weighted average-based recommendation method to recommend items.

**NUSCCF [16]:** NUSCCF is a fast method for searching neighbor users in the INT, NINU and UNI item subspaces. NUSCCF defines the novel similarity to predict rating, which is related to the neighbor trees constructed in three subspaces.

**SENU-ACPC:** SENUS-ACPC is the combination of SENUS and AC-PCC. For a target user, SENUS is used for searching its neighbor users, and the similarity measure defined in AC-PCC is used for predicting ratings of recommended items.

**SENU-NHSM:** SENUS-NHSM is the combination of SENUS and NHSM. We first adopt SENUS to find neighbor users for a target user, and then use the similarity measure defined in NHSM to predict ratings of recommended items.

**SENU-BCF:** SENUS-BCF is the combination of SENUS and BCF. SENUS is used in the stage of searching, and BCF the stage of prediction.

**SENU-HUSM:** SENUS-HUSM is the combination of SENUS and HUSM, where SENUS searches neighbor users, and HUSM predicts ratings.

Most methods mentioned above have one main parameter  $k$ , the number of neighbor users. In ACC-PC, NHSM, BCF and HUSM, we empirically set the number of neighbor users to 20.

Each of datasets in experiments is categorized into five subsets by applying five-fold cross-validation, which follows the way in [15] and [29]. In each trial, four subsets are used for training and the remaining for test. As a result, we report the average performance on five trials. To evaluate the recommendation quality of compared algorithms, we adopt three performance indexes, mean absolute error (MAE), recall, and coverage [24, 27, 32]. MAE is the most commonly used metric for measuring the accuracy of a recommendation algorithm, and is defined by comparing the predicted rating values against actual rating values. Namely,

$$MAE = \frac{1}{m'} \sum_{a=1}^{m'} \frac{1}{n'_a} \sum_{j=1}^{n'_a} |r_{aj} - \hat{r}_{aj}| \quad (12)$$

where  $m'$  is the number of users in the test set and  $n'_a$  is the number of all predicted items for the target user  $u_a$ ,  $r_{aj}$  and  $\hat{r}_{aj}$  are the actual and predicted ratings of user  $u_a$  on item  $t_j$ , respectively. Note that a smaller MAE indicates a better performance. Recall is defined as below:

$$Recall = \frac{1}{m'} \sum_{a=1}^{m'} \frac{|IR_{ap} \cap IR_{aa}|}{|IR_{aa}|} \quad (13)$$

where  $IR_{ap}$  represents the predicted recommendation list (set) for user  $u_a$ , and  $IR_{aa}$  is the set of actual recommendation items for user  $u_a$  in the test set. A greater Recall means better performance. Coverage is a measure of the percentage of items for which a RS can provide recommendations [27], and is expressed as follows:

$$Coverage = \frac{\sum_{a=1}^{m'} |IR_{ap} \cap T_{at}|}{\sum_{a=1}^{m'} |T_{at}|} \quad (14)$$

where  $T_{at}$  denotes the set of items rated by user  $u_a$  in the test set.

### 3.2 Experiments on ML-Latest-Small

The ML-Latest-Small dataset is from MovieLens which was collected by the GroupLens Research Project at the University of Minnesota [12]. This dataset describes 5-star rating that movies were rated on a floating point scale of 0.5 (bad) to 5 (excellent) with scale of 0.5, and contains 100,004 ratings across 9,066 movies. The ML-Latest-Small dataset was created by 671 users from January 09, 1995 to October 16, 2016. Users were selected at random for inclusion. All selected users had rated at least 20 movies. The sparsity of ML-Latest-Small is 98.3%.

SENU has the parameter  $k$ . To illustrate the effect of the parameter  $k$  on the prediction performance of SENUS-based methods, we perform 5-fold cross validation on the ML-Latest-Small dataset. Let  $k$  vary in the range of  $\{5, \dots, 15\}$ . The variation of average performance indexes vs.  $k$  on the prediction performance of SENUS-based methods is given in Fig. 3, where Figs. 3(d), (a), (c), and (b) show SENUS-NHSM, SENUS-ACPCC, SENUS-BCF, and SENUS-HUSM, respectively. These figures are dual ordinate coordinate diagrams where MAE, Recall, and Coverage curves are on the left, and Running time is on the right.

Observation on the curves of Running time indicates that the running time has an approximately linear relationship with  $k$ . The greater  $k$  is, the more time SENUS-based methods need. Thus,  $k$  should not be too greater. From curves of three performance indexes of MAE, Recall, and Coverage, we can see a commonality of them. These indexes fluctuate so little as  $k$  increases so that there is no significant change from the curves. Even so, it is better to set  $k$  between 6 and 9 according to the comparative experimental results. In SENUS-based methods, we set the neighbor parameter  $k = 8$ .

Table 1 shows the performance comparison on the ML-Latest-Small dataset, including average MAE, Recall, Coverage and Running time. From Table 1, we can have the following conclusions.

- On the index of Running time, SENUS-ACPCC is the fastest and followed by SENUS-NHSM. However, the fast algorithm UTAOS ranks the third. In other words, our proposed scheme may have a faster speed than the existing fast methods which do not adopt hybrid similarity measures and have bad recommendation performance. Note that the running time of SENUS-based

methods depends on the hybrid similarity measure in the prediction stage. Since the computational complexity of HUSM is relatively great, SENUS-HUSM takes more time.

- It can be seen that SENUS-based methods are apparently superior to the fast searching methods UTAOS and NUSCCF on the performance of MAE, Recall, and Coverage, which indicates that our SENUS-based methods can enhance the recommendation performance of fast algorithms.
- Compared to the original algorithms, SENUS-based methods are faster and have a compared performance. For example, SENUS-HUSM is almost three times faster than HUSM on the index of Running time. The MAE obtained by SENUS-HUSM is slightly better than that of HUSM.

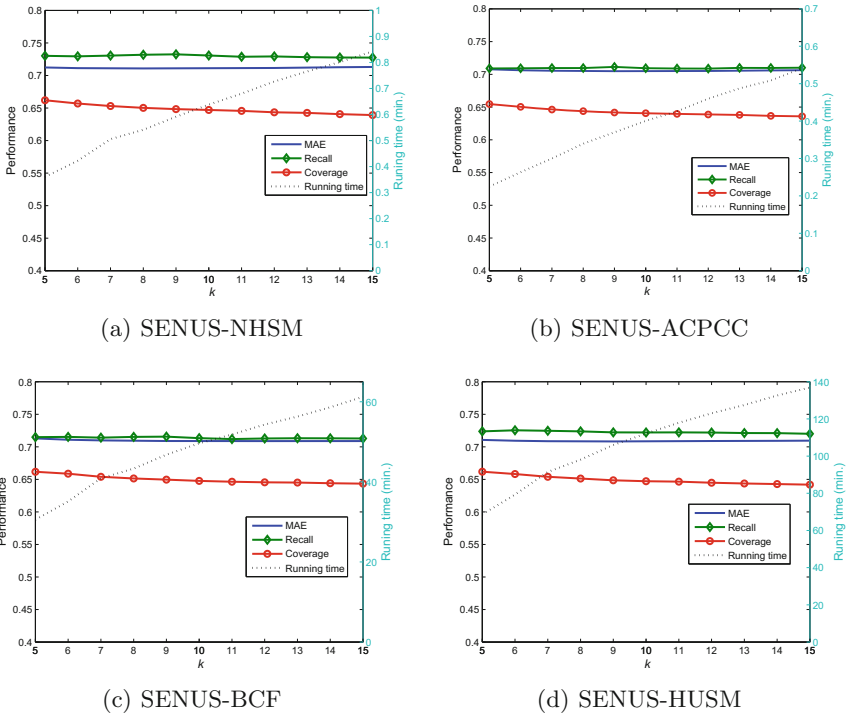


Fig. 3. Comparison of performance indexes vs.  $k$  in SENUS-based methods

### 3.3 Experiments on ML-100K

The ML-100k dataset is also from MovieLens [12]. The data was collected through the MovieLens web site ([movielens.umn.edu](http://movielens.umn.edu)) and has been cleaned up, where users who had less than 20 ratings or did not have complete demographic

information were removed from this data set. This data set contains 100,000 ratings provided by 943 users for 1,682 movies. Movies were rated on an integer scale 1 (bad) to 5 (excellent). This matrix has a sparsity of 93.7%.

**Table 1.** Performance comparison on the ML-Latest-Small dataset

Method	MAE	Recall	Coverage	Running time (min.)
UTAOS	0.8021	0.5495	0.3758	0.6740
NUSCCF	1.0880	0.079	0.1682	4.9166
NHSM	0.7128	0.7244	0.6600	3.2
SENU-NHSM	0.7107	0.7319	0.6502	0.5414
ACPCC	0.6983	0.7345	0.6593	5.81
SENU-ACPCC	0.7049	0.7096	0.6437	0.3395
BCF	0.7087	0.7248	0.6545	181.8
SENU-BCF	0.7092	0.7152	0.6513	43.30
HUSM	0.7102	0.7211	0.6520	380
SENU-HUSM	0.7083	0.7237	0.6512	98

**Table 2.** Comparison of six methods on the ML-100K dataset

Method	MAE	Recall	Coverage	Running time (min.)
UTAOS	0.8638	0.4393	0.336	0.2481
NUSCCF	0.9808	0.1248	0.1615	5.3972
NHSM	0.7424	0.6398	0.5947	4.3
SENU-NHSM	0.7379	0.6896	0.6207	0.5257
ACPCC	0.7294	0.6636	0.6054	5.76
SENU-ACPCC	0.7366	0.6691	0.6156	0.2389
BCF	0.7452	0.6834	0.6308	188.19
SENU-BCF	0.7423	0.6793	0.6261	18.2105
HUSM	0.7453	0.6672	0.6141	377.6
SENU-HUSM	0.7405	0.6813	0.6226	37.2075

Table 2 gives the comparison of ten methods on the ML-100K dataset. We have a similar conclusion as before. The MAE, Recall and Coverage of SENUS-based methods are much better than both UTAOS and NUSCCF, and very close to the corresponding original methods. On the index of Running time, SENUS-based methods are faster than the corresponding original methods, and may be faster than UTAOS or NUSCCF.

## 4 Conclusions

Neighbor-based collaborative filtering algorithms rely on the historical behavior of neighboring users, and recommend the target users based on historical data of neighbors. Thus, finding the effective neighbor users and providing higher accuracy in RSs are the major issues. This paper proposes an SENSUS method for searching the effective neighbor users with little running time consume. SENSUS first calculates the co-rating support of pairwise users in the INT, NINU, and UNI item subspaces, respectively and then combines them to generate the total co-rating supports of pairwise users, which are used to find neighbor users for a target user. SENSUS can be combined with any similarity measure which is adopted in the prediction stage. Four SENSUS-based methods are designed, or SENSUS-ACPCC, SENSUS-NHSM, SENSUS-BCF and SENSUS-NUSM.

Experimental results on the ML-Latest-Small and ML-100K datasets indicate that SENSUS can greatly speed up the recommendation procedure of the original NCF methods. In other words, SENSUS is effective when searching neighbor users. Moreover, compared with the other fast methods UTAOS and NUSCCF, SENSUS-based methods have an obvious advantage in the recommendation performance of MAE, Recall, and Coverage. Since the running time of SENSUS-based methods mainly depends on the complexity of similarity measures, some SENSUS-based methods can have a fast recommendation procedure than the existing fast methods.

## References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.* **17**(6), 734–749 (2005)
2. Agarwal, N., Haque, E., Liu, H., Parsons, L.: Research paper recommender systems: a subspace clustering approach. In: Fan, W., Wu, Z., Yang, J. (eds.) *WAIM 2005*. LNCS, vol. 3739, pp. 475–491. Springer, Heidelberg (2005). [https://doi.org/10.1007/11563952\\_42](https://doi.org/10.1007/11563952_42)
3. Ahn, H.J.: A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Inf. Sci.* **178**(1), 37–51 (2008)
4. Bilge, A., Polat, H.: A comparison of clustering-based privacy-preserving collaborative filtering schemes. *Appl. Soft Comput.* **13**(5), 2478–2489 (2013)
5. Birtolo, C., Ronca, D.: Advances in clustering collaborative filtering by means of fuzzy C-means and trust. *Expert Syst. Appl.* **40**(17), 6997–7009 (2013)
6. Bobadilla, J., Ortega, F., Hernando, A.: Recommender systems survey. *Knowl. Based Syst.* **46**(1), 109–132 (2013)
7. Bobadilla, J., Ortega, F., Hernando, A.: A collaborative filtering similarity measure based on singularities. *Inf. Process. Manage.* **48**(2), 204–217 (2012)
8. Choi, S.M., Ko, S.K., Han, Y.S.: A movie recommendation algorithm based on genre correlations. *Expert Syst. Appl.* **39**(9), 8079–8085 (2012)
9. Cleger-Tamayo, S., Fernández-Luna, J.M., Huete, J.F.: Top-n news recommendations in digital newspapers. *Knowl. Based Syst.* **27**(6), 180–189 (2012)
10. Garcia, I., Sebastia, L., Onaindia, E.: On the design of individual and group recommender systems for tourism. *Expert Syst. Appl.* **38**(6), 7683–7692 (2011)

11. Guan, Y., Zhao, D., Zeng, A., Shang, M.S.: Preference of online users and personalized recommendations. *Physica A Stat. Mech. Appl.* **392**(16), 3417–3423 (2013)
12. Harper, F.M., Konstan, J.A.: The movielens datasets: history and context. *ACM Trans. Interact. Intell. Syst.* **5**(4), 1–19 (2015)
13. Hu, R., Dou, W., Liu, J.: ClubCF: a clustering-based collaborative filtering approach for big data application. *IEEE Trans. Emerg. Topics Comput.* **2**(3), 302–313 (2014)
14. Phelps, J.E., Lewis, R., Mobilio, L.J., Perry, D.K., Raman, N.: Viral marketing or electronic word-of-mouth advertising: examining consumer responses and motivations to pass along email. *J. Advertising Res.* **44**(44), 333–348 (2009)
15. Koochi, H., Kiani, K.: User based collaborative filtering using fuzzy C-means. *Measurement* **91**, 134–139 (2016)
16. Koochi, H., Kiani, K.: A new method to find neighbor users that improves the performance of collaborative filtering. *Expert Syst. Appl.* **83**, 30–39 (2017)
17. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. *Computer* **42**(8), 30–37 (2009)
18. Koutrika, G., Bercovitz, B., Garcia-Molina, H.: FlexRecs: expressing and combining flexible recommendations. In: *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, SIGMOD 2009*, no. 14, pp. 745–758. ACM, New York (2009)
19. Li, Q., Myaeng, S.H., Kim, B.M.: A probabilistic music recommender considering user opinions and audio features. *Inf. Process. Manage.* **43**(2), 473–487 (2007)
20. Li, T., Ding, C.: The relationships among various nonnegative matrix factorization methods for clustering. In: *Sixth International Conference on Data Mining (ICDM 2006)*, pp. 362–371, December 2006
21. Liu, H., Hu, Z., Mian, A., Tian, H., Zhu, X.: A new user similarity model to improve the accuracy of collaborative filtering. *Knowl. Based Syst.* **56**(3), 156–166 (2014)
22. Martinez-Cruz, C., Porcel, C., Bernabé-Moreno, J., Herrera-Viedma, E.: A model to represent users trust in recommender systems using ontologies and fuzzy linguistic modeling. *Inf. Sci.* **311**, 102–118 (2015)
23. Patra, B.K., Launonen, R., Ollikainen, V., Nandi, S.: Exploiting Bhattacharyya similarity measure to diminish user cold-start problem in sparse data. In: Džeroski, S., Panov, P., Kocev, D., Todorovski, L. (eds.) *Discovery Science*, pp. 252–263. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-11812-3\\_22](https://doi.org/10.1007/978-3-319-11812-3_22)
24. Patra, B.K., Launonen, R., Ollikainen, V., Nandi, S.: A new similarity measure using bhattacharyya coefficient for collaborative filtering in sparse data. *Knowl. Based Syst.* **82**, 163–177 (2015). <http://www.sciencedirect.com/science/article/pii/S0950705115000830>
25. Pirasteh, P., Hwang, D., Jung, J.E.: Weighted similarity schemes for high scalability in user-based collaborative filtering. *Mobile Netw. Appl.* **20**(4), 497–507 (2015)
26. Ramezani, M., Moradi, P., Akhlaghian, F.: A pattern mining approach to enhance the accuracy of collaborative filtering in sparse data domains. *Physica A Stat. Mech. Appl.* **408**(32), 72–84 (2014)
27. Sarwar, B.M., Konstan, J.A., Borchers, A., Herlocker, J., Miller, B., Riedl, J.: Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. In: *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work, CSCW 1998*, no. 10, pp. 345–354. ACM, New York (1998)
28. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. *Adv. Artif. Intell.* **2009**(1), 1–19 (2009)

29. Tsai, C.F., Hung, C.: Cluster ensembles in collaborative filtering recommendation. *Appl. Soft Comput. J.* **12**(4), 1417–1425 (2012)
30. Wang, Y., Deng, J., Gao, J., Zhang, P.: A hybrid user similarity model for collaborative filtering. *Inf. Sci.* **418–419**, 102–118 (2017)
31. Wen, H., Fang, L., Guan, L.: A hybrid approach for personalized recommendation of news on the web. *Expert Syst. Appl.* **39**(5), 5806–5814 (2012)
32. Zhang, J., Peng, Q., Sun, S., Liu, C.: Collaborative filtering recommendation algorithm based on user preference derived from item domain features. *Physica A Stat. Mech. Appl.* **396**(2), 66–76 (2014)





# Towards both Local and Global Query Result Diversification

Ming Zhong<sup>1</sup>(✉), Huanyu Cheng<sup>1</sup>, Ying Wang<sup>2</sup>, Yuanyuan Zhu<sup>1</sup>,  
Tieyun Qian<sup>1</sup>, and Jianxin Li<sup>3</sup>

<sup>1</sup> School of Computer Science, Wuhan University, Wuhan 430072, China  
{clock, chy, yzhu, qty}@whu.edu.cn

<sup>2</sup> Guotai Junan Securities Co. Ltd., Shanghai 201201, China  
wangying020826@gtjas.com

<sup>3</sup> School of Information Technology, Deakin University,  
Burwood, VIC 3125, Australia  
jianxin.li@deakin.edu.au

**Abstract.** Query result diversification is critical for improving users' query satisfaction by making the top ranked results cover more different query semantics. The state-of-the-art works address the problem via bi-criteria (namely, relevance and dissimilarity) optimization. However, such works only consider how dissimilar the returned results are to each other, which is referred to "local diversity". In contrast, some works consider how similar the not returned results are to the returned results, which is referred to "global diversity", and however need a user defined threshold to predicate whether a result set is diverse. In this paper, we extend the traditional bi-criteria optimization problem to a tri-criteria problem that considers both local diversity and global diversity. For that, we formally define the metrics of global diversity and global-and-local diversity. Then, we prove the NP-hardness of the proposed problems, and propose two heuristic algorithms, greedy search and vertex substitution, and sophisticated optimization techniques to solve the problems efficiently. To evaluate our approach, we perform comprehensive experiments on three real datasets. The results demonstrate that our approach can indeed find more reasonably diversified results. Moreover, our greedy search algorithm can significantly reduce the time cost by leveraging the critical object, and then our vertex substitution algorithm can incrementally improve the objective value of results returned by greedy search with extra time cost.

**Keywords:** Query result diversification · Algorithm · Optimization

## 1 Introduction

Traditionally, the results of information retrieval or database query are only ranked by their own features, such as relevance to the query, authority, etc. It

The original version of this chapter was revised: The acknowledgement section was updated. The correction to this chapter is available at [https://doi.org/10.1007/978-3-030-18579-4\\_46](https://doi.org/10.1007/978-3-030-18579-4_46)

may cause that the top ranked results are too homogeneous to meet different users' demands. For example, given an ambiguous keyword like "apple", if all the top ranked results are about the Apple company, such as its official website, logo, founder and products, it obviously reduces the query satisfaction of users who want the information about apple as a fruit. Thus, query result diversification is getting more and more attention for improving users' satisfaction, and has been widely used in many applications [1–3, 5–8, 10, 11, 13, 15].

As summarized by the survey [16], most previous works [9, 13] use the dissimilarity (also known as distance) between each pair of results in a specific  $K$ -size candidate result set as a metric of its diversity. Since this kind of diversity is determined only by the  $K$  candidates, we call it *local diversity*. While, some other works [3, 5, 7, 15] measure diversity from a different perspective. Typically, they aim to find a  $K$ -size set of results such that the other results not belong to the set are similar to at least one of them, namely, can be "covered" by them. We call this kind of diversity that is determined by the dissimilarity between the candidates and the other results as *global diversity*. Both the local diversity and the global diversity are to make the returned  $K$  results include as most different interpretation of query as possible.

Moreover, there are usually two ways of importing the diversity into query result evaluation, no matter local or global. One is *predication*, which means a set of candidate results is either diverse, if the dissimilarity between results can meet a certain condition, or not diverse, otherwise. For instance, a set of candidate results is generally considered to be diverse if the dissimilarity between each pair of candidates is greater than a given threshold, in the case of local diversity. The other is *bi-criteria*, which combines the own feature of each candidate and the diversity of the set of  $K$  candidates, which is evaluated as the aggregation of dissimilarity between results, into a unified objective function. For instance, the objective function of the classical diversification algorithm MMR (Maximal Marginal Relevance) [4] is the linear summation of candidates' relevance and the minimum dissimilarity between each pair of candidates.

The predication style approaches [5, 7–9, 12–15] that focus on finding the most relevant diverse result set may suffer from evaluating the threshold. Since the degree of dissimilarity between results depends on the given query, it is actually infeasible to diversify the results of different queries with a fixed threshold. Even if we can tune the threshold to adapt to different queries, how to get a suitable threshold for a particular query is still intractable. In contrast, the bi-criteria style approaches [2, 4, 9] try to optimize the objective value of returned result set without the threshold. Due to the NP-hardness of such optimization problems, the heuristic or approximate algorithms are usually applied. However, the existing bi-criteria style approaches only consider the local diversity. Although some works have studied the problem of global diversification, they only apply the predication style approach. For example, DISC [7] proposes the zoom-in and zoom-out operators to adjust the dissimilarity threshold for achieving better global diversity. To the best of our knowledge, the bi-criteria style approach

that aims to find the globally diverse or even both locally and globally diverse result set has not been well studied yet.

Therefore, we extend the bi-criteria approach to a novel tri-criteria approach in this paper, which leverages the global diversity to improve query result diversification. Firstly, we formally define the global diversity and global-and-local diversity for the bi-criteria style approach. Let  $P$  be the set of candidate results to a given query. For a subset of candidate results  $S \subseteq P$  with  $|S| = K$ , the global diversity is the maximum dissimilarity between any result in  $P \setminus S$  and its most similar result in  $S$ . Then, the global-and-local diversity integrates the aggregated dissimilarity between the  $K$  candidates in  $S$  together. As a result, the tri-criteria approach that applies global-and-local diversification will return more reasonably diversified relevant results.

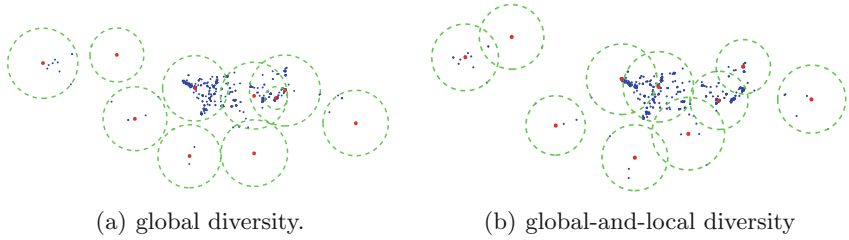
For example, we visualize a set of candidate results as dots according to their dissimilarity/distance to each other in a 2D space in Fig. 1. The red dots are the ten results returned by our diversification approach. The green circles are used to visualize the coverage of returned results (i.e., red dots as centers), and their radius are actually their maximum dissimilarity to the blue dots that are more similar to them than other red dots. There is a red dot with a very small circle that is too close to two other red dots in Fig. 1(a), because only global diversity is considered. Thus, many blue dots in the middle of the space are far away from the red dots. In contrast, we can see that the distribution of red dots chosen by global-and-local diversification in Fig. 1(b) is more reasonable.

To address the problem of tri-criteria approach, we propose two heuristic algorithms. The first one adopts a greedy methodology like the classical bi-criteria approach MMR, namely, chooses the result with the maximum marginal gain and adds it to the final result set iteratively. Moreover, a critical pruning technique is also proposed to improve the efficiency. The second one is a vertex substitution algorithm that optimizes the returned result set of the first algorithm by substituting one of its results with another result not belong to the final result set iteratively. For finding the optimal substitution quickly, we present a dedicated algorithm and the corresponding data structures.

Our contributions are summarized as follows.

- We firstly define global diversity and global-and-local diversity for bi-criteria approach, and formalize a novel tri-criteria problem.
- We propose two heuristic algorithms, greedy search and vertex substitution, and several optimization techniques to solve the problem.
- We perform comprehensive experiments on three real datasets. The experimental results show that global-and-local diversification is more effective than local diversification and global diversification. Moreover, our algorithms and optimization techniques can improve the efficiency significantly.

The rest of this paper is organized as follows. Section 2 introduces the formulation of problem. Section 3 introduces the greedy search algorithm. Section 4 introduces the vertex substitution algorithm. Section 5 introduces the experiments. Lastly, Sect. 6 concludes the paper.



**Fig. 1.** A comparison between global diversification and global-and-local diversification. (Color figure online)

## 2 Problem Formulation

Let  $P$  be a set of objects matched by a user query and  $S \subset P$  with  $|S| = K$  be the set of objects returned by diversification from  $P$ . We denote by  $o_i \in S$  and  $p_i \in P \setminus S$  an object chosen as the diversified result and an object not chosen as the diversified result, respectively. We refer to  $o_i \in S$  as a diverse object.

We measure the dissimilarity between two objects using a user-defined distance function  $dis : P \times P \mapsto [0, 1]$ . The specific definition of  $dis$  depends on the applications. For example, Jaccard distance based on label and Cosine distance based on TF/IDF are often used for information retrieval. In particular, we use  $N(o_i) \subset P \setminus S$  to denote the set of neighbors of an object  $o_i \in S$ , each of which is an object  $p_i \in P \setminus S$  that is closer to  $o_i$  than any other object in  $S \setminus o_i$ . We call  $o_i$  the center of  $p_i \in N(o_i)$ . The formal definition of  $N(o_i)$  is given as follows.

$$N(o_i) = \{p_i | dis(p_i, o_i) \leq dis(p_i, o_j), \forall o_j \in S \wedge o_j \neq o_i\} \quad (1)$$

Given a set of neighbors  $N(o_i)$  of  $o_i$ , we denote by  $r_{o_i}$  the maximum distance between  $o_i$  and each neighbor  $p_i \in N(o_i)$ , and by  $r$  the maximum value of  $r_{o_i}$  for all  $o_i \in S$ . Thus we have

$$r = \max_{o_i \in S} r_{o_i} = \max_{o_i \in S} \max_{p_i \in N(o_i)} dis(o_i, p_i) \quad (2)$$

Moreover, we denote by  $d$  the minimum distance between each pair of objects in  $S$ . Thus we have

$$d = \min_{o_i, o_j \in S, o_i \neq o_j} dis(o_i, o_j) \quad (3)$$

The  $r$  and  $d$  are actually the metrics of global diversity and local diversity respectively. For a diversified result set, the shorter the  $r$  the better the global diversity, and the longer the  $d$  the better the local diversity. In addition, we denote by  $weight_{o_i} \in [0, 1]$  the relevance of  $o_i$  to the query. Then we define the optimal result set  $S_g$  with respect to global diversity and the the optimal result set  $S_{g+l}$  with respect to global-and-local diversity as follows.

**Definition 1** ( $S_g$ ). Given a set of objects  $P$  matched by a given query and an integer  $K \geq 1$ , a set of objects  $S_g$  is optimal with respect to global diversity if and only if

1.  $S_g \subset P$ ;
2.  $|S_g| = K$ ;
3.  $f_g = \mu \sum_{o_i \in S_g} weight_{o_i} / K - (1 - \mu)r$  is maximized.

where  $\mu \in (0, 1)$  is the weight of relevance.

With the objective function  $f_g$ , the objects in  $S_g$  should be as relevant to the query as possible and also as similar to the other objects in  $P \setminus S_g$  as possible. Thereby, each object in  $P \setminus S_g$  can have at least a relevant representative that is close to it in semantics in  $S_g$ .

**Definition 2** ( $S_{g+l}$ ). Given a set of objects  $P$  matched by a given query and an integer  $K \geq 1$ , a set of objects  $S_{g+l}$  is optimal with respect to global-and-local diversity if and only if

1.  $S_{g+l} \subset P$ ;
2.  $|S_{g+l}| = K$ ;
3.  $f_{g+l} = \mu \frac{1}{K} \sum_{o_i \in S_g} weight_{o_i} - (1 - \mu)(r - d)$  is maximized;

where  $\mu \in (0, 1)$  is the weight of relevance.

The difference between the objective functions of  $S_{g+l}$  and  $S_g$  is that the objects in  $S_{g+l}$  should also be as dissimilar to each other as possible. Therefore,  $S_{g+l}$  is optimal with respect to tri-criteria.

Lastly, we define the two problems to be addressed in this paper. The first one is global diversification, and the second one is global-and-local diversification.

*Problem 1.* Given a set of objects  $P$  and an integer  $K \geq 1$ , find the optimal subset  $S_g \subset P$  with respect to global diversity.

*Problem 2.* Given a set of objects  $P$  and an integer  $K \geq 1$ , find the optimal subset  $S_{g+l} \subset P$  with respect to global-and-local diversity.

In order to find an exact solution to the above problems, we need to traverse the whole solution space, the size of which is  $C_{|P|}^K$ . Moreover, evaluating the objective function for each solution costs  $O(K|P|)$  time according to Definitions 1 and 2. Thus, the total time complexity of the proposed problems is  $O(\frac{K|P|*|P|!}{(K-1)!})$ .

### 3 Greedy Search Algorithm

We present a greedy search algorithm to address the proposed problems in this section. The basic greedy algorithm iteratively selects the optimal object into the result set, which maximizes the objective value of the current set, until there

have been  $K$  objects in the set, so that it can obtain a relatively reliable solution to the problems avoiding traversing the whole solution space.

From Eq. (2), we can know for each diverse object  $o_i$  selected into  $S$ , the basic greedy algorithm has to traverse all  $p_i \in P \setminus S$  to calculate the maximum distance  $r$ . In order to further improve the efficiency of the greedy algorithm, we propose a pruning method to avoid repeatedly calculating the maximum distance  $r$  of the objective function. The pruning method is based on two properties we observe. To demonstrate the properties, we need the following two definitions. We denote by  $center(p)$  the nearest diverse object  $o$  of an object  $p$ . The critical object denoted as  $*p$  is defined as followed:

$$*p = \arg \max_{p_i \in P \setminus S} r \quad (4)$$

*Property 1.* If we add an diverse object to  $S$ , the  $r$  of  $S$  decreases or does not change.

*Proof.* First, if the critical object  $*p$  does not change when a new diverse object is added, no matter its center diverse object, denoted by  $center(*p)$ , changes or not, it is easy to know that  $dis(*p, center(*p)) \leq r$ . Second, if  $*p$  changes, we denote by  $*p_{new}$  the new critical object. If its new center diverse object  $center_{new}(*p_{new})$  is the new diverse object  $o_{new}$  added to  $S$ , we denote the previous center of  $*p_{new}$  by  $center_{pre}(*p_{new})$ . We have  $dis(*p_{new}, o_{new}) < dis(*p_{new}, center_{pre}(*p_{new}))$ . Because  $dis(*p_{new}, center_{pre}(*p_{new})) \leq r$ , we have  $dis(*p_{new}, o_{new}) < r$  which means  $r$  decreases. If  $center_{new}(*p_{new})$  is the diverse object denoted by  $o_{pre}$  in previous  $S$ , we have  $dis(*p_{new}, o_{pre}) \leq r$  which means the  $r$  does not change or decrease. In summary, we prove the property.

*Property 2.* If  $r$  decreases due to adding a new object to  $S$ , the new diverse object  $o_{new}$  must be closer to the critical object  $*p$  than the previous center diverse object of  $*p$ .

*Proof.* Assume that when  $r$  decreases, the new diverse object  $o_{new}$  is not closer to  $*p$  than its previous center  $center_{pre}(*p)$ . We can deduce that the center of  $*p$  does not change. First, if  $*p$  is still the critical object,  $r$  does not change. This contradicts the condition that  $r$  decreases. Second, we denote the new critical object by  $*p_{new}$ ,  $*p_{new} \neq *p$ . If its new center diverse object  $center_{new}(*p_{new})$  is the new diverse object  $o_{new}$  added to  $S$ , we denote the previous center of  $*p_{new}$  by  $center_{pre}(*p_{new})$ . We have  $dis(*p_{new}, o_{new}) < dis(*p_{new}, center_{pre}(*p_{new}))$ . Since  $dis(*p_{new}, center_{pre}(*p_{new})) \leq dis(*p, center_{pre}(*p_{new}))$ , we deduce that  $dis(*p_{new}, o_{new}) \leq dis(*p, center_{pre}(*p_{new}))$ . However, since  $*p_{new}$  is the critical object, we have  $dis(*p_{new}, o_{new}) \geq dis(*p, center_{pre}(*p_{new}))$ . They are paradoxical. If its new center diverse object  $center_{new}(*p_{new})$  is the diverse object  $o_{pre}$  in previous  $S$ , we have  $dis(*p_{new}, o_{pre}) \leq dis(*p, center_{pre}(*p))$ . This also contradicts that  $*p_{new}$  is the critical object. So the assumption does not hold. The property has been proved.

**Algorithm 1.** Greedy Search.

---

**Require:** a matched object set  $P$ , a real positive number  $K$   
**Ensure:** a final result set  $S$

```

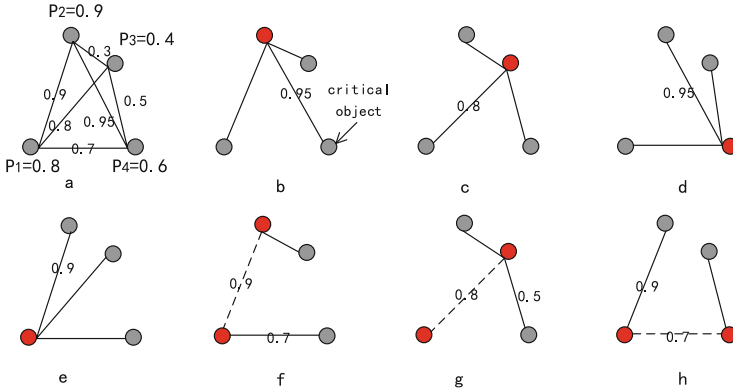
1:  $candidateSet \leftarrow P$ ,  $topSet \leftarrow \emptyset$ ,  $r \leftarrow \infty$ 
2: for  $i = 1$  to  $K$  do
3:   for all  $p \in candidateSet$  do
4:     if  $|topSet| == 0$  then
5:       for all  $p' \in candidateSet$ ,  $p' \neq p$  do
6:         if  $dis(p, p') > r$  then
7:            $r \leftarrow dis(p, p')$ 
8:            $*p' \leftarrow p'$ 
9:         end if
10:      end for
11:     else if  $dis(p, *p) < dis(C(*p), *p)$  then
12:       for all  $p' \in candidateSet$  do
13:          $tempR \leftarrow \min\{dis(p', C(p')), dis(p', p)\}$ 
14:         if  $tempR > r$  then
15:            $r \leftarrow tempR$ 
16:            $*p' \leftarrow p'$ 
17:         end if
18:       end for
19:     else
20:        $r \leftarrow dis(C(*p), *p)$ 
21:     end if
22:     calculate the function score  $f(p)$ 
23:     if  $f(p) > f(o)$  then
24:        $o \leftarrow p$ ,  $f(o) \leftarrow f(p)$ 
25:     end if
26:   end for
27:    $topSet.add(o)$ ,  $candidateSet.remove(o)$ 
28:   update the current critical object  $*p$ ,  $*p \leftarrow *p'$ 
29:   update the index  $C$  and  $T$ 
30: end for
31:  $S \leftarrow topSet$ 
32: return  $S$ 

```

---

With Properties 1 and 2, we can know that if the new diverse object is not closer to the critical object  $*p$  than the previous center diverse object of  $*p$ , the  $r$  of  $S$  must not change. Therefore, we can avoid recalculating  $r$  for some new diverse objects which are not closer to the critical object  $*p$  than the previous center diverse object of  $*p$ . This greatly improves the efficiency of the algorithm when  $|P|$  is particularly large since it costs  $O(|P|)$  to calculate  $r$ . The worst time complexity of the algorithm for Problem 2 is  $O(K|P|^2)$ . The average time complexity for Problem 2 is close to that for Problem 1, namely,  $O(K|P|)$ .

Algorithm 1 gives the pseudo code of our greedy algorithm. In the Algorithm 1 we use some indexes to reduce the time complexity.  $C$  is an index which records the nearest diverse object  $o$  of each object  $p$ , and  $T$  records the nearest other diverse object of each diverse object  $o$ .  $topSet$  is used to save the currently selected diverse object and  $candidateSet$  is used to save the other objects. The algorithm selects the diverse object into  $topSet$  one by one until the  $|topSet|$  reaches  $K$ . At the beginning, when there is no object in  $topSet$ , the algorithm adds the object farthest from other objects (line 4–10). In each iteration, the algorithm firstly decides whether the new diverse object is closer to the critical object  $*p$  than the previous center diverse object of  $*p$  (line 11). If true, the algorithm retrieves  $r$  by traversing all object in  $candidateSet$  (line 12–18). Otherwise,  $r$  is



**Fig. 2.** An example of greedy-based search. (Color figure online)

equal to the distance between the critical object  $*p$  and the center diverse object of  $*p$ , namely,  $dis(C(*p), *p)$  (line 20). Then, the algorithm calculates the function score and selects the object  $o$  into  $topSet$  which makes the function score  $f$  maximal (line 22–27). At the end of iteration, the algorithm updates the critical object  $*p$  and the index  $T$  and  $C$  of which time complexity is  $O(|K|)$  and  $O(|P|)$  (line 28–29). Finally, a final result subset  $S$  will be returned until it loops  $K$  times (line 32).

**Example 1.** Figure 2 shows an example of greedy based search. There are four objects in  $P = \{p_1, p_2, p_3, p_4\}$ . The distance between objects in  $P$  and the weight of each object are shown in Fig. 2(a). In the Fig. 2(b–h), the red point is the object that has been selected to be a diverse object temporarily. The critical object is the object furthest from the diverse object. For example, in Fig. 2(b),  $p_2$  is the diverse object and  $p_4$  is the corresponding critical object which determines that  $r = 0.95$ . Assume  $\mu = 0.3$ . We have  $f_g(b) = f_{g+l}(b) = -0.395$ ,  $f_g(c) = f_{g+l}(c) = -0.44$ ,  $f_g(d) = f_{g+l}(d) = -0.485$ ,  $f_g(e) = f_{g+l}(e) = -0.39$ . So after the first iteration,  $p_1$  is picked into the  $topSet$ . In the next iteration,  $f_g(f) = -0.235$ ,  $f_g(g) = -0.17$ ,  $f_g(h) = -0.42$  and  $f_{g+l}(f) = 0.395$ ,  $f_{g+l}(g) = 0.39$ ,  $f_{g+l}(h) = 0.07$ . For the Problem 1, the  $p_3$  is picked and for the Problem 2, the  $p_2$  is picked.

## 4 Vertex Substitution Algorithm

Let  $X = \{o_1, o_2, \dots, o_K\}$  denote a feasible solution of the problem we defined. The *vertex substitution* neighborhood of  $X$ , denoted as  $Nor(X)$ , is obtained by replacing, in turn, each diverse object belonging to the  $topSet$  by each object in the  $candidateSet$ . Thus, the cardinality of  $Nor(X)$  is  $|K| * (|P| - |K|)$ . For a certain object  $o \in candidateSet$ ,  $Nor_o(X)$  is obtained by replacing each diverse object belonging to the  $topSet$  by the object  $o$ . The cardinality of  $Nor_o(X)$  is  $|K|$ . Our approach finds the best solution  $X' \in Nor_o(X)$ . If  $f_g(X') > f_g(X)$  or



**Algorithm 2.** Vertex Substitution.

---

**Require:** a initial result set  $topSet$ , a candidate object set  $candidateSet$   
**Ensure:** a final result set  $S$

```

1: while  $\exists p \in candidateSet, \exists o \in topSet, f(p) > f(o)$  do
2:   for all  $in \in candidateSet$  do
3:      $out \leftarrow \emptyset$ 
4:      $rin \leftarrow 0, Ro \leftarrow \emptyset, Rout \leftarrow \emptyset$ 
5:     for all  $p \in candidateSet$  do
6:       if  $dis(p, in) < dis(p, C(p))$  then
7:         if  $dis(p, in) > rin$  then
8:            $rin = dis(p, in)$ 
9:         end if
10:      else
11:        if  $dis(p, C(p)) > Ro(C(p))$  then
12:           $Ro(C(p)) = dis(p, C(p))$ 
13:        end if
14:         $Rout(C(p)) = \max\{Rout(C(p)), \min\{dis(p, in), dis(p, SC(p))\}\}$ 
15:      end if
16:    end for
17:     $ro1 = \max_{o \in topSet} Ro(o)$ 
18:     $co = \arg \max_{o \in topSet} Ro(o)$ 
19:     $ro2 = \max_{o' \in topSet \& o' \neq co} Ro(o')$ 
20:    for all  $o \in topSet$  do
21:      if  $o == co$  then
22:         $r = \max\{rin, ro2, Rout(o)\}$ 
23:      else
24:         $r = \max\{rin, ro1, Rout(o)\}$ 
25:      end if
26:      calculate the current function score  $f(o)$ 
27:      if  $f(o) > f_{max}$  then
28:         $out \leftarrow obj, f_{max} \leftarrow f(o)$ 
29:      end if
30:    end for
31:    if  $f$  increase then
32:       $candidateSet.remove(in), candidate.add(out)$ 
33:       $topSet.remove(out), topSet.add(in)$ 
34:    end if
35:  end for
36:  update the index  $C, SC$  and  $T$ 
37: end while
38:  $S \leftarrow topSet$ 
39: return  $S$ 

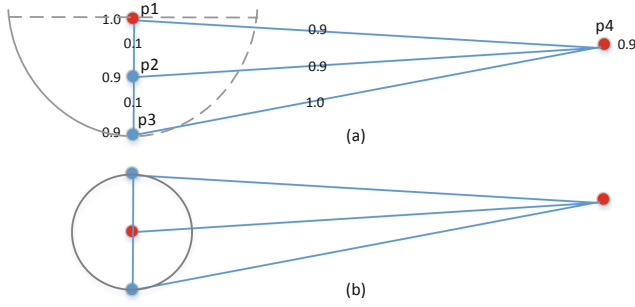
```

---

$f_{g+l}(X') > f_{g+l}(X)$ , a replacement is made ( $X \leftarrow X'$ ), a new neighborhood  $Nor_{new}(X)$  is defined, and the process is repeated. Otherwise, continue to try the next object in  $candidateSet$ . If the solution cannot be improved eventually, the procedure stops. Since it costs too much time to calculate  $r$ , we apply a method to quickly retrieve  $r$ . The pseudo codes is given in the Algorithm 2.

Compared to greedy based search algorithm, it can usually gives a better solution with higher  $f_g$  or  $f_{g+l}$ . Experiments show that it can usually improve the result of greedy based algorithm when it uses the solution of the greedy based algorithm as the initial solution.

**Example 2.** Figure 3 shows an example of improving the solution of greedy based algorithm through vertex substitution. There are four objects in  $P = \{p_1, p_2, p_3, p_4\}$  with their corresponding weight and the distance between each other. Figure 3(a) is the solution of the greedy based search algorithm. The  $topSet$  is composed of red points. Figure 3(b) is the solution that replacing  $p_1$



**Fig. 3.** Improve the solution of greedy based algorithm through vertex substitution.

with  $p2$  into the  $topSet$ . Assume  $\mu = 0.5$ . We have  $f_g(a) = 0.375, f_{g+l}(a) = 0.825$  and  $f_g(b) = 0.4, f_{g+l}(b) = 0.85$ . Obviously Fig. 3(b) is a better solution whether it is Problems 1 or 2.

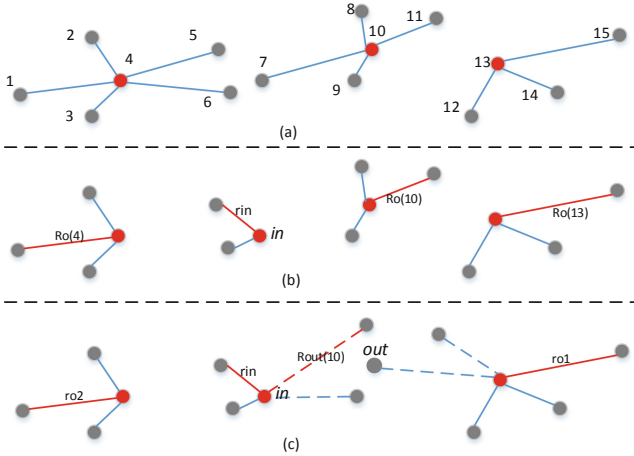
In the Algorithm 2, there are  $K$  possible replacements by replacing each diverse object in  $topSet$  by this object. We will find the best solution  $X'$  and if  $f_g(X') > f_g(X)$  or  $f_{g+l}(X') > f_{g+l}(X)$  ( $X$  is the original solution), a replacement is made. In order to calculate the  $f$  of each solution, we use the following data structure shown in Table 1.

**Table 1.** Data structure.

Character	Meaning
$SC(p)$	the second nearest diverse object $o$ of an object $p \in candidateSet$
$in$	the new diverse object added to the $topSet$
$rin$	the radius of the new diverse object $in$
$Ro(o)$	the current radius of the original diverse object $o$ when $in$ is added to $topSet$
$co$	$\arg \max_{o \in topSet} Ro(o)$
$ro1$	$\max_{o \in topSet} Ro(o)$
$ro2$	$\max_{o' \in topSet \& o' \neq co} Ro(o')$
$out$	the diverse object that is selected to be deleted from $topSet$
$Rout(o)$	$\max_{p \in N(o) \cup o} dis(p, C(p))$ , where object $o$ is selected as $out$

Algorithm 2 will continually replace the result in  $topSet$  until the function score  $f$  cannot increase. Firstly, the algorithm updates the radius  $rin$  of the new diverse object  $in$ , the current radius  $Ro(o)$  of all original diverse object  $o$  when  $in$  is added to  $topSet$  and the radius  $Rout(o)$  if  $o$  is removed from  $topSet$  (line 5–16). Then, it finds the maximum value  $ro1$  from  $Ro(o)$  and the corresponding object  $co$  (line 17–18). What’s more, it finds the second maximum value  $ro2$  (line 19). Thus, we can avoid repeating traversing all object in  $candidateSet$  to calculate  $r$  in the following replacement. The algorithm traverses the diverse object  $o$  in

*topSet* to find the object *out* that the function score  $f$  is the maximum if *out* is removed from the *topSet* according to whether *o* is *co* (line 20–30). At the end of iteration, if the function score is greater than the origin score, the new diverse object *in* will be added to *topSet* and *out* will be removed (line 31–34). Before determining whether the function score  $f$  is the maximum, it updates the index  $C, SC$  and  $T$  (line 36). Finally, the algorithm returns the final result set  $S$  (line 39). The time complexity is  $O(|P|^2)$ .



**Fig. 4.** Example of replacement.

**Example 3.** Figure 4 shows a simple example to quickly calculate  $r$  with the data structure  $rin, Ro(), ro1, ro2, Rout()$ . Figure 4(a) shows that there are 15 objects in  $P$ , and the  $topSet = \{4, 10, 13\}$ . In the Fig. 4(b), object 7 is added to the  $topSet$ .  $rin$  is the largest distance between *in* and the object it covered.  $Ro()$  records the largest distance between the diverse object and the object it covered. In the Fig. 4(c),  $Rout()$  records the largest distance between an object and its currently center diverse object when its original center diverse object is deleted from  $topSet$ . According to the  $Ro()$ , we can find the  $ro1$  and  $ro2$ . As shown in the Fig. 4(c),  $out \neq 13$ , thus,  $r = \max\{ro1, rin, Rout(out)\}$ . Otherwise, if  $out == 13$ ,  $r = \max\{ro2, rin, Rout(out)\}$ .

## 5 Experiments

In this section we evaluate the performance of our proposed algorithms using real datasets. The experiments are performed on a Windows 2012 server with 3.3 GHz CPU and 128 GB memory. Our algorithms are implemented in Java 1.7.

**Datasets and Algorithms.** We evaluate the performance of our algorithms with varied parameters using two real point datasets. Moreover, a real document

dataset is used to evaluate the performance of our algorithms on real document search. The two real point datasets, “Brightkite” and “Gowalla”<sup>1</sup>, are the collection of 2-dimensional points representing geographic information. We randomly and uniformly assign weights to objects in real point datasets which are normalized in  $[0,1]$ . We use the normalized Euclidean distance which can be seen as dissimilarity for real point datasets. The real document dataset, “Reuters”<sup>2</sup>, contains 21,578 news assigned with one or multiple tags. We measure the similarity of documents using the TF\*IDF score normalized by the length of the corresponding document.

Table 2 summarizes the participant algorithms. The first four are greedy search algorithms to solve Problems 1 or 2. The difference is whether the algorithm applies critical object  $*p$  for optimization. The next four are vertex substitution algorithms to solve Problems 1 or 2 with different initial solution. The last one is the classic local diversification algorithm MMR. The parameter  $\mu$  is set to 0.2 in all the following experiments.

**Table 2.** Participant algorithms.

Algorithm	Abbreviation	Description
Greedy-Basic-Problem1	GBP1	Do not apply critical object $*p$ for optimization
Greedy-Basic-Problem2	GBP2	Do not apply critical object $*p$ for optimization
Greedy-Optimized-Problem1	GOP1	Apply critical object $*p$ for optimization
Greedy-Optimized-Problem2	GOP2	Apply critical object $*p$ for optimization
Substitution-RandomInitialValue-Problem1	SRP1	The initial solution is randomly assigned
Substitution-RandomInitialValue-Problem2	SRP2	The initial solution is randomly assigned
Substitution-GreedyInitialvalue-Problem1	SGP1	Apply the solution of greedy algorithm as the initial solution
Substitution-GreedyInitialvalue-Problem2	SGP2	Apply the solution of greedy algorithm as the initial solution
Maximal-Marginal-Revenue	MMR	Algorithm based on maximal marginal revenue

**Exp-1.** In this experiment we test the effectiveness of our optimization strategy in the greedy-based search algorithm on both point datasets. In the previous section, we mention that the critical object  $*p$  can be used to save much overhead because adding an object into the *topSet* far away from the critical object do not

<sup>1</sup> <https://snap.stanford.edu/data>.

<sup>2</sup> <http://kdd.ics.uci.edu/databases/reuters21578/>.

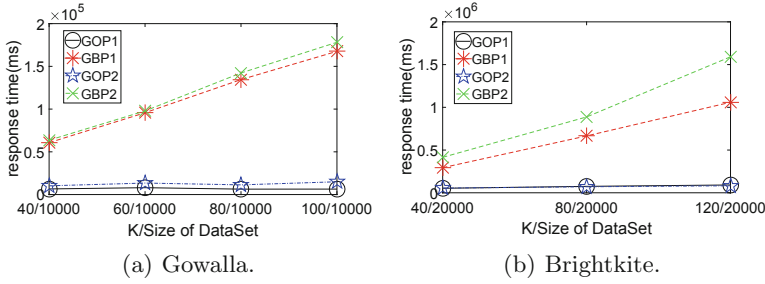


Fig. 5. Test optimization strategy of greedy-based search.

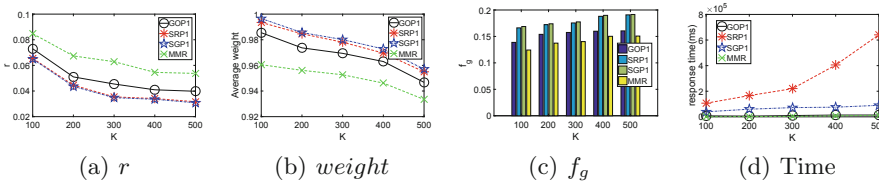


Fig. 6. Problem 1: Gowalla.

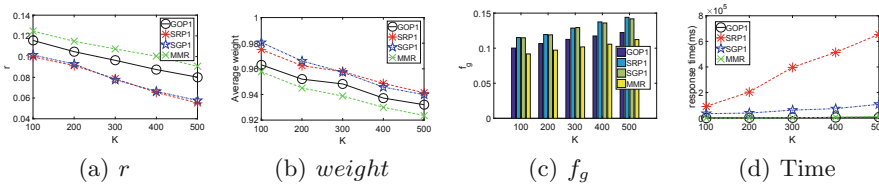


Fig. 7. Problem 1: Brightkite.

change  $r$ . The test results are shown in Fig. 5 and all the experimental results demonstrated in the following are actually the average of 50 sets of experiments.

In Fig. 5(a) and (b), we set the dataset size to 10000 and 20000 respectively. When  $K$  increases, the response time of all algorithms increases. GBP1 and GBP2 increase sharply and GOP1 and GOP2 keep stable. In Fig. 5(a), when  $K$  increases to 100, the response time of GBP1 is 26 times that of GOP1 and GBP2 is 12 times that of GOP2. In Fig. 5(b), when  $K$  increases to 120, the multiples are 11 and 19 respectively. GOP1 and GOP2 take seconds or tens of seconds while GBP1 and GBP2 may take a few minutes. This proves that our optimization strategy can effectively reduce the time overhead of the algorithm when  $K$  increases.

**Exp-2.** In this experiment we test three algorithms GOP1, SRP1 and SGP1 for Problem 1. We also get the local-diversified results obtained by the method MMR under the same conditions. We compare the results of the four algorithms on  $r$ ,  $weight$ , objective function value  $f_g$  and average response time of diverse

objects on both point datasets. The dataset size is set to 5000 and all the experimental results demonstrated in the following are actually the average of 50 sets of experiments.

Figures 6 and 7 show the testing results with varied  $K$ . Figures 6(a) and 7(a) show the comparison of  $r$  of the four algorithms. Algorithm MMR is the worst for the optimization of  $r$ . Because the objective function of MMR focus on optimizing the distance between diverse objects and *weight* of diverse objects. This causes that there is an object not similar to any diverse objects in the return set. Thus, the return set may lack representativeness. Algorithms SRP1 and SGP1 perform best. In Figs. 6(b) and 7(b), for optimizing *weight* of diverse objects, GOP1, SRP1 and SGP1 also perform better than MMR. When  $K$  increases,  $r$  and *weight* of diverse objects decrease. This shows that a user can get more representative results by increasing  $K$  although it may introduce some less relevant results into the return set.

From Figs. 6(c) and 7(c), we can see that when we use the objective function of global-diversity to measure the results of local-diversity, the value of MMR is the worst obviously. Figures 6(d) and 7(d) show the response time of the four algorithms. GOP1 and MMR perform best. Algorithm SGP1 is much faster than SRP1. The difference between the algorithm SGP1 and SRP1 is that the former applies the solution of greedy-based algorithm as the initial solution and the initial solution of the latter is randomly assigned. Although SGP1 costs the time of both greedy-based algorithm and vertex substitution-based algorithm, it saves much more time by applying less vertex substitution. Since the greedy-based algorithm has little time overhead compared with the vertex substitution-based algorithm, applying the vertex substitution-based algorithm to optimize the solution of the greedy-based algorithm is a good choice.

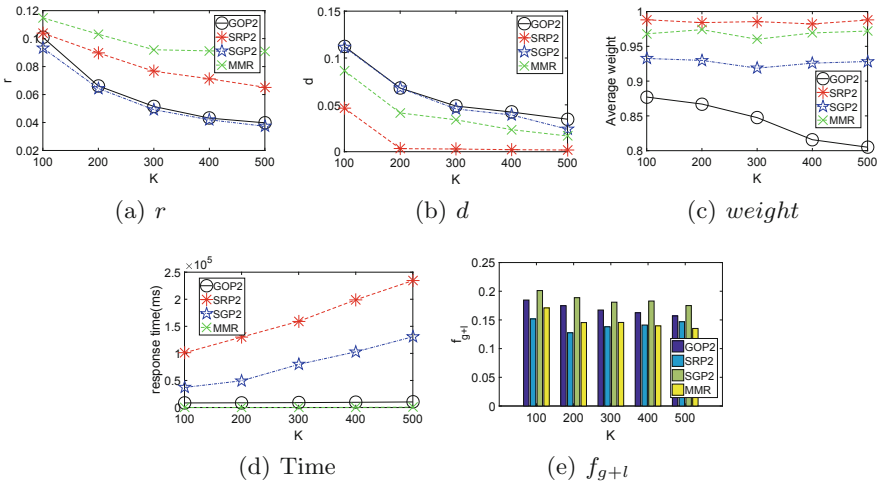


Fig. 8. Problem 2: Gowalla.

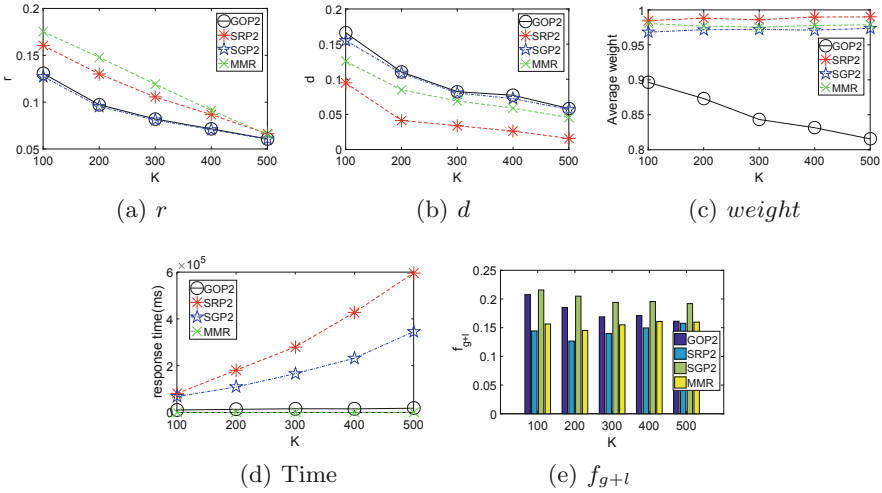


Fig. 9. Problem 2: Brightkite.

**Exp-3.** In this experiment we test three algorithms GOP2, SRP2 and SGP2 for Problem 2. Here we also compare our results with the results obtained by MMR. We compare the results of the four algorithms on  $r$ ,  $d$ ,  $weight$ , objective function value  $f_{g+l}$  and average response time of diverse objects on both point datasets. The dataset size is set to 5000 and all the experimental results demonstrated in the following are actually the average results of 50 sets of experiments.

Figures 8 and 9 show the testing results with varied  $K$ . We have the following observations.

- In Figs.8(a) and 9(a), when  $K$  increases,  $r$  decreases. However, in the Brightkite dataset, as  $K$  increases,  $r$  of all four algorithms eventually converges. In the Gowalla dataset, algorithm SGP2 and GOP2 always perform best and MMR perform worst. This may be because the objects of the Brightkite dataset are distributed more evenly in space.
- In Figs.8(b) and 9(b), algorithm SRP2 has a poor performance, especially when  $K$  increases. It shows that the randomized initial solution is not good for vertex substitution-based algorithm to solve the Problem 2. In Figs.8(c) and 9(c), when  $K$  increases, the average weight of SGP2, SRP2 and MMR keeps stable and the average weight of algorithm GOP2 decreases. From Figs.8(e) and 9(e), we can conclude that SGP2 is currently the best way to solve Problem 2. In summary, SRP2 performs worst for solving the Problem 2 and it is even worse than MMR. Thus it is important and vital to give a relatively good initial solution for the vertex substitution-based algorithm.
- In Figs.8(d) and 9(d), when  $K$  increases, the response time of GOP2 and MMR keeps stable while the response time of SRP2 and SGP2 increases. Similar to Problem 1, although SGP2 costs the time of both greedy-based algorithm and vertex substitution-based algorithm, it still saves much more

**Table 3.** Results of keyword search results on Reuters.

Keyword	All tag number	K	Tag number		
			MMR	SGP1	SGP2
oper	3	3	2	2	3
		7	3	2	3
		16	3	3	3
payout	15	26	11	4	15
		238	15	13	15
		260	15	15	15
china	91	10	11	7	19
		50	33	15	53
		100	50	36	78
paris	145	50	44	42	59
		100	77	57	103
		150	108	89	118
today	322	100	76	73	80
		200	124	104	140
		300	145	136	164

time by applying less number of vertex substitution. In summary, applying good initial solution to the vertex substitution-based algorithm can not only improve the value of the objective function, but also effectively reduce the response time.

**Exp-4.** In this experiment we evaluate the effectiveness of algorithms for respectively traditional local diversity, Problems 1 and 2, namely, global diversity and global and local diversity in real document dataset “Reuters”. From the above experiments, we can find that SGP1 and SGP2 are the best solutions to Problems 1 and 2. Thus, we compare the representative local method MMR and the best solution to Problems 1 and 2, i.e. SGP1 and SGP2 in keyword search.

Table 3 shows the performance of MMR, SGP1 and SGP2 on keyword search of documents. From the table, we have the following observation. When we select a special keyword such as “oper” which only corresponds to three tags, namely, “canada”, “earn” and “usa”, SGP2 can obtain the results of all tags with  $K$  set to 3 while MMR and SGP1 can only obtain the results with two tags. When  $K$  is set larger as to 7, MMR can obtain all tags. SGP1, by contrast, has to set  $K$  to be 16 until it obtains all tags. Likewise, keyword “payout” is the same. We find that  $K$  has to be set terribly large with not a lot overall tags for MMR and SGP1 if we want to get the results of all tags. Thus, we can conclude that SGP2 can return the least results with all tags. In other words, SGP2 can cover all keyword semantics with least results.



For a great many overall tags of keyword such as “china”, “paris” and “today”, we compare the tag number of different algorithms with varied  $K$ . We find that SGP2 always returns results with more tags at the same  $K$ . Although in experiment 2 SGP1 performs better than MMR, in this experiment MMR takes a second place while SGP1 is the worst. Because the solution to Problem 1 can only find the representative results, which cannot ensure that the returned results are dissimilar. By contrast, the algorithm for local diversity is to find the different results as far as possible, which can include more tags easily. Thus, we can conclude that SGP2 can contain more tags or semantics in results of the same number. Overall, the algorithm SGP2 integrating both local and global diversity can find more different and representative results.

## 6 Conclusion

In this paper, we study a novel tri-criteria optimization problem that aims to find a subset of  $K$  query results with the optimal linear summation of relevance, local diversity and global diversity. We formalize the problem and present two heuristic algorithms with sophisticated optimization techniques to address it. We perform experiments on real datasets. Compared with the previous bi-criteria approaches that only consider the local diversity, our approach can improve the global diversity of final results with an extra time cost, which can be reduced remarkably by our algorithms. Consequently, we establish an extended general query result diversification framework.

**Acknowledgement.** This paper was supported by National Natural Science Foundation of China under Grant No. 61202036, 61502349 and 61572376 and Natural Science Foundation of Hubei Province under Grant No. 2018CFB616.

## References


1. Agrawal, R., Gollapudi, S., Halverson, A., Jeong, S.: Diversifying search results. In: WSDM, pp. 5–14 (2009)
2. Angel, A., Koudas, N.: Efficient diversity-aware search. In: SIGMOD, pp. 781–792 (2011)
3. Capannini, G., Nardini, F.M., Perego, R., Silvestri, F.: Efficient diversification of web search results. VLDB **4**(7), 451–459 (2011)
4. Carbinell, J., Goldstein, J.: The use of MMR, diversity-based reranking for reordering documents and producing summaries. SIGIR **51**(2), 335–336 (1998)
5. Demidova, E., Fankhauser, P., Zhou, X., Nejdl, W.: DivQ: diversification for keyword search over structured databases. In: SIGIR, pp. 331–338 (2010)
6. Deng, T., Fan, W.: On the complexity of query result diversification. ACM Trans. Database Syst. **39**(2), 15 (2014)
7. Drosou, M., Pitoura, E.: Disc diversity: result diversification based on dissimilarity and coverage. VLDB **6**(1), 13–24 (2012)
8. Fraternali, P., Martinenghi, D., Tagliasacchi, M.: Top-k bounded diversification. In: SIGMOD, pp. 421–432 (2012)

9. Gollapudi, S., Sharma, A.: An axiomatic approach for result diversification. In: WWW, pp. 381–390 (2009)
10. Hu, S., Dou, Z., Wang, X., Sakai, T., Wen, J.: Search result diversification based on hierarchical intents. In: CIKM, pp. 63–72 (2015)
11. Liu, Z., Sun, P., Chen, Y.: Structured search result differentiation. VLDB **2**(1), 313–324 (2009)
12. Qin, L., Yu, J.X., Chang, L.: Diversifying top-k results. VLDB **5**(11), 1124–1135 (2012)
13. Vee, E., Shanmugasundaram, J., Amer-Yahia, S.: Efficient computation of diverse query results. IEEE Data Eng. Bull. **32**(4), 57–64 (2009)
14. Vieira, M.R., et al.: On query result diversification. In: ICDE, pp. 1163–1174 (2011)
15. Zhao, F., Zhang, X., Tung, A.K.H., Chen, G.: Broad: diversified keyword search in databases. VLDB **4**(12), 1355–1358 (2011)
16. Zheng, K., Wang, H., Qi, Z., Li, J., Gao, H.: A survey of query result diversification. Knowl. Inf. Syst. **51**(1), 1–36 (2017)

# **Social Network**



# Structured Spectral Clustering of PurTree Data

Xiaojun Chen<sup>1</sup> , Chao Guo<sup>1</sup>, Yixiang Fang<sup>2</sup>, and Rui Mao<sup>1</sup>

<sup>1</sup> College of Computer Science and Software, Shenzhen University, Shenzhen,  
People's Republic of China

{xjchen,mao}@szu.edu.cn, guo28296@vip.qq.com

<sup>2</sup> Department of Computer Science, University of Hong Kong,  
Pok Fu Lam, Hong Kong  
yxfang@cs.hku.hk

**Abstract.** Recently, a “Purchase Tree” data structure is proposed to compress the customer transaction data and a local PurTree Spectral clustering method is proposed to recover the cluster structure from the purchase trees. However, in the PurTree distance, the node weights for the children nodes of a parent node are set as equal and the difference between different nodes are not distinguished. In this paper, we propose a Structured PurTree Subspace Spectral (SPSS) clustering algorithm for PurTree Data. In the new method, we propose a PurTree subspace similarity to compute the similarity between two trees, in which a set of sparse and structured node weights are introduced to distinguish the importance of different nodes in a purchase tree. A new clustering model is proposed to learn a structured graph with explicit cluster structure. An iterative optimization algorithm is proposed to simultaneously learn the structured graph and node weights. We propose a balanced cover tree for fast k-NN searching during building affinity matrices. SPSS was compared with six clustering algorithms on 10 benchmark data sets and the experimental results show the superiority of the new method.

**Keywords:** Transaction data · Clustering · Structure clustering

## 1 Introduction

Transaction data is the collection of daily shopping transactions of customers at a retail company. A transaction is a sequence of products (items) bought by a customer in one basket. Clustering of customer transaction data is one of the most critical tasks in successful modern marketing and customer relationship management [9]. It is used to categorize customers into different groups based on their purchase behaviors, so that the customers in the same cluster bought more similar products to each other than to those in other clusters. The early segmentation methods use general variables like customer demographics, lifestyle, attitude and psychology, because such variables are intuitive and easy to operate [6]. With

the rapid increase of customer behavior data, the new study turns to use product specific variables such as items purchased [7, 12, 16]. Although some methods were proposed to segment customers based on the item set data [13, 15, 16], the huge amount of item sets hinders the usage of these methods in real applications.

Recently, Chen et al. proposed a PurTreeClust clustering algorithm for customer segmentation from large-scale transaction data [2, 3]. In this algorithm, a product tree is used to organize the categories in the transaction data, in which the leaf nodes denote products and the internal nodes represent product categories. A purchase tree is built for each customer, in which each purchased product corresponds to a leaf node in the product tree. A PurTree distance metric was defined to measure the difference between two purchase trees, and a PurTreeClust method was proposed to cluster purchase trees. The PurTreeClust algorithm first builds a cover tree for indexing the purchase tree data set, then selects initial cluster centers with a fast density estimation method. Finally, the clustering result is obtained by assigning each customer to its nearest cluster center. Chen et al. further proposed a Local PurTree Spectral clustering method, which can learn an adaptive similarity matrix from the local distances and the level weights in the PurTree distance simultaneously [4]. In their method, the node weights in the PurTree distance are set as equal for the children nodes of a parent node and a parameter  $\gamma$  is used to control the level weights. However, since a product tree often consists of hundreds of thousands nodes, it is desired to learn a set of sparse node weights such that only a few important nodes are considered.

In this paper, we propose a clustering algorithm to address the above problems, namely Structured PurTree Subspace Spectral (SPSS). A PurTree subspace similarity is proposed to compute the similarity between two trees, in which a set of sparse node weights are introduced to distinguish the importance of different nodes in a purchase tree. The new model aims to learn a structured graph that best approximates the input node affinity matrices and contains explicit cluster structure. We present an iterative optimization algorithm to optimize the proposed model, in which the structured graph and node weights are simultaneously learned. A balanced cover tree is proposed for fast k-NN searching during building affinity matrices. A series of experiments were conducted on 10 benchmark data sets and the experimental results show the superior performance of SPSS.

The rest of this paper is organized as follows. Notations and preliminaries are given in Sect. 2. We propose the PurTree subspace distance in Sect. 3, and the SPSS algorithm in Sect. 5. The experimental results and analysis are presented in Sect. 6. Conclusions and future work are given in Sect. 7.

## 2 Notations

Let  $T$  be a tree with nodes  $N(T)$  and edges  $E(T) \in N(T) \times N(T)$ . The root of  $T$  is denoted by  $root(T)$ . A node without children is a leaf, and otherwise an internal node. For an edge  $\{v, w\} \in E(T)$ , node  $v$  is the parent and  $w$  is a

child,  $P_w(T) = v$  and  $w \in C_w(T)$ . The descendants of  $v$ , the nodes in all paths reachable from  $v$  to a leaf node, is denoted as  $des(v)$ . The level of a node is defined by  $1 +$  (the number of edges between the node and the root).  $N^l(T)$  represents nodes in the  $l$ -th level of  $T$ . The height of tree  $T$ , denoted by  $H(T)$ , is the number of edges on the longest downward path between the root and a leaf node.

Let  $\Psi$  be a rooted tree used to systematically organize the items with multiple levels of categories, in which each leaf node represents an item and each internal node represents a category. All leaf nodes in  $\Psi$  are assumed to have equal depth in [3]. A purchase tree  $\varphi$  is used to illustrate the items bought by a customer, which is a subgraph of  $\Psi$ , i.e.,  $N(\varphi) \subseteq N(\Psi)$ ,  $E(\varphi) \subseteq E(\Psi)$ . Given a leaf node  $u \in N(\varphi)$ , the path from  $root(\varphi)$  to  $u$  also exists in  $\Psi$ . For each purchase tree  $\varphi$ , we have  $H(\varphi) = H(\Psi)$ .

### 3 PurTree Subspace Similarity

Given a product tree  $\Psi$  and a set of  $n$  purchase trees  $\Phi = \{\varphi_1, \dots, \varphi_n\}$  where  $\varphi_i \in \Psi$ , let  $H(\Phi)$  be the height of these purchase trees,  $root(\varphi)$  be the empty root node of the purchase tree. The PurTree distance is defined as follows [3]

$$d(\varphi_i, \varphi_j) = \sum_{l=1}^{H(\Phi)} \beta_l \sum_{v \in N^l(\varphi_i) \cup N^l(\varphi_j)} \alpha_v \delta_v(\varphi_i, \varphi_j) \tag{1}$$

where  $\delta_v(\varphi_i, \varphi_j)$  is the Jaccard distance of  $\varphi_i$  and  $\varphi_j$  on an internal node  $v$ , which is defined as

$$\delta_v(\varphi_i, \varphi_j) = 1 - \frac{|C_v(\varphi_i) \cap C_v(\varphi_j)|}{|C_v(\varphi_i) \cup C_v(\varphi_j)|} \tag{2}$$

and  $\alpha_v$  is the node weight for node  $v \in N(\Psi)$  and  $\beta_l$  is the  $l$ -th level weight.  $\{\beta_1, \dots, \beta_{H(\Phi)}\}$  is a geometric sequence with common ratio  $\gamma$  ( $\gamma \geq 0$ ) under constraint  $\sum_{l=1}^{H(\Phi)} \beta_l = 1$ . In 2017, Chen et al. proposed a spectral clustering method LPS to learn a set of optimal level weights  $\beta$  for the PurTree distance [4]. However, the node weights  $\alpha$  are still set to fixed values.

In the PurTree distance [3], the node weights  $\alpha$  are set as fixed values and a parameter  $\gamma$  is used to control the level weights  $\beta$ . However, since a product tree often consists of hundreds of thousands nodes, it is desired to learn a set of sparse node weights such that only a few important nodes are considered. Moreover, most graph-based clustering algorithms require an affinity matrix  $\mathbf{A} \in \mathbb{R}^{n \times n}$  as input. In this paper, we propose an approach to construct initial affinity graph  $\mathbf{A}$ . Given a set of  $n$  purchase trees  $\Phi = \{\varphi_1, \dots, \varphi_n\}$  and a set of node weights  $\Omega = \{\omega_v | v \in N(\Phi)\}$ , we define the similarity between two purchase trees  $\varphi_i$  and  $\varphi_j$  as

$$a_{ij} = \sum_{l=1}^{H(\Phi)} \sum_{v \in N^l(\varphi_i) \cup N^l(\varphi_j)} \omega_v a_{ij}^v \tag{3}$$

where  $a_{ij}^v$  is the node similarity between  $\varphi_i$  and  $\varphi_j$  on node  $v$ .  $\Omega = \{\omega_v | v \in N(\Phi)\}$  consists of the node weights for the purchase tree data set  $\Phi$ . To well model the tree structure in  $\Phi$ , we impose the following constraint on  $\Omega$

$$\begin{cases} \forall v \in N(\Psi), \omega_v \in [0, 1] \\ \sum_{l=1}^{H(\Phi)} \sum_{v \in N^l(\Phi)} \omega_v = 1 \\ \sum_{t \in C_v(\Psi)} \omega_t = \omega_v \end{cases} \tag{4}$$

It is desirable to learn the affinity values of  $\mathbf{A}$  such that smaller node distance  $\delta_v(\varphi_i, \varphi_j)$  corresponds to a larger node similarity  $s_{ij}^v$ . In addition, we simply set  $a_{ii} = 0$ . For each  $\mathbf{a}_i$ , we propose to learn  $s_{ij}^v$  by solving the following problem

$$\min_{\mathbf{a}_i^T \mathbf{1}=1, \mathbf{a}_i \geq 0, a_{ii}=0} \sum_{j=1}^n \sum_{l=1}^{H(\Phi)} \sum_{v \in N^l(\varphi_i) \cup N^l(\varphi_j)} \delta_v(\varphi_i, \varphi_j) a_{ij}^v + \gamma \sum_{j=1}^n \sum_{l=1}^{H(\Phi)} \sum_{v \in N^l(\varphi_i) \cup N^l(\varphi_j)} (a_{ij}^v)^2 \tag{5}$$

which can be rewritten as

$$\min_{\mathbf{a}_i^T \mathbf{1}=1, \mathbf{a}_i \geq 0, a_{ii}=0} \sum_{l=1}^{H(\Phi)} \sum_{v \in N^l(\varphi_i) \cup N^l(\varphi_j)} \left\| \mathbf{a}_i^v + \frac{\delta_i^v}{2\gamma} \right\|_2^2 \tag{6}$$

where  $\mathbf{a}_i^v \in \mathbb{R}^{n \times 1}$ ,  $\delta_i^v \in \mathbb{R}^{n \times 1}$  consists of  $\{\delta_v(\varphi_i, \varphi_1), \dots, \delta_v(\varphi_i, \varphi_n)\}$ .

In real applications, we often prefer a sparse affinity matrix  $\mathbf{A}$ . To achieve this goal, we can learn  $\mathbf{A}$  with the maximal  $\gamma$  such that the optimal solution  $\mathbf{a}_i^v$  to problem (5) has exactly  $k$  nonzero values. Let  $\{\delta_{i1}^v, \dots, \delta_{in}^v\}$  is ordered from small to large as  $\{d_{i,1}^v, \dots, d_{i,n}^v\}$  and  $\mathcal{M}_k(\varphi_i)$  contains the  $k$ -nearest-neighbors of  $\varphi_i$  according to  $\{\delta_{i1}^v, \dots, \delta_{in}^v\}$ . Here,  $d_{i,i}^v$  is set as  $+\infty$  in order to make  $a_{ii}^v = 0$ . With a similar procedure as those in [11], we obtain the optimal affinities  $a_{ij}^v$  as follows

$$a_{ij}^v = \begin{cases} \frac{d_{i,k+1}^v - \delta_{ij}^v}{k d_{i,k+1}^v - \sum_{l=1}^k d_{i,l}^v} & \varphi_j \in \mathcal{M}_k(\varphi_i) \\ 0 & otherwise \end{cases} \tag{7}$$

## 4 Fast $k$ -NN Search with Balanced Cover Tree

In this section, we propose a balanced cover tree for fast  $k$ -nearest neighbor searching of purchase trees during building affinity matrices.

### 4.1 Balanced Cover Tree

Cover tree, first invented by Beygelzimer et al., is a leveled tree data structure for fast nearest neighbor operations in metric spaces [1]. In this paper, we extend the original cover tree to a balanced cover tree.

Let  $CT$  be a cover tree on a data set  $S$  with base parameter  $\alpha$ . Each level in  $CT$  is indexed by an integer scale  $l$  which decreases from top to bottom. Let the cover set  $CS_l$  denote the set of objects in  $S$  associated with the nodes at level  $l$ . Each object in  $S$  appears at most once in each level, but may be associated

with multiple nodes in the tree. The basic operation to build a cover tree is the insertion operation, which is shown in Algorithm 1. Here,  $Q_l$  is a subset of the objects at level  $l$  which may contain the new object  $p$  as a descendant. If  $S$  consists of  $n$  objects, the computational complexity of inserting an object is  $O(c^6 \log(n))$  [1]. A cover tree can be built on a data set  $S$  by sequentially applying Algorithm 1 to insert each object in  $S$ .

In the original paper [1], the base parameters  $\alpha$  is fixed as 2. In this paper, we set the base parameter  $\alpha \in [1.62, +\infty]$  in order to build more balanced cover tree for fast  $k$ -nearest neighbor searching. The following theorem ensures the correctness of Algorithm 1.

---

**Algorithm 1.** Insert (object  $p$ , cover set  $Q_l$ , level  $l$ , base parameter  $\alpha$ )

---

**Output:** **true** if insertion succeeds, **false** otherwise.

```

1: Set  $Q_{l-1} = \{q \in Q_l : d(p, q) \leq \alpha^l\}$ .
2: if  $Q_{l-1} = \emptyset$  then
3:   return false.
4: else
5:   if Insert( $p, Q_{l-1}, l - 1$ ) fails then
6:     Pick  $q = \arg \min_{q \in Q_l} d(p, q)$ .
7:     if  $d(q, p) = 0$  then
8:       exit.
9:     else
10:      insert  $p$  as a child of  $q$ .
11:      return true.
12:    end if
13:  else
14:    return true.
15:  end if
16: end if

```

---

**Theorem 1.** Given a cover tree  $CT$  on  $S$ ,  $Insert(p, CS_\infty, \infty, \alpha)$  returns a cover tree on  $S \cup \{p\}$  if  $\alpha \geq \frac{1+\sqrt{5}}{2}$ .

*Proof.* If there exists an object  $q \in Q_l$  such that  $d(p, q) = 0$ , Algorithm 1 may be infinitely called. To avoid such infinite looping, we add step 6 to drop such object  $p$ . If  $\forall q \in CT, d(q, p) > 0$ , we can easily prove that  $p$  is guaranteed to be insert into some level in  $CT$ . The nesting and covering tree invariants can be proved in a similar way as those for Theorem 4 in [1].

We now prove that the insertion maintains separation invariant. If  $p$  is inserted in level  $l - 1$ , consider  $q \in CS_{l-1}$ . If  $q \in Q = \{C(q) | q \in Q_l\}$ ,  $p$  can be inserted into  $l - 1$  level iff  $Q_{l-1} = \emptyset$ , which means that  $d(p, q) > \alpha^{l-1}$ . If  $q \notin Q = \{C(q) | q \in Q_l\}$ , there must exist some parent of  $q$  in iteration  $l' > l$ , say  $q' \in CS_{l'-1}$ , was eliminated in step 1, which implies that  $d(p, q') > \alpha^{l'}$ . Assume that there exists a set of  $\{q_j | l - 1 \leq j \leq l' - 1\}$  such that  $q_{j+1}$  is the parent



of  $q_j, q_{l-1} = q'$  and  $q_{l-1} = p$ . Using the covering tree invariant and triangular inequality, we have

$$\begin{aligned} d(p, q) &\geq d(p, q') - \sum_{j=l-1}^{l'-2} d(q_j, q_{j+1}) > d(p, q') - \sum_{j=l}^{l'-1} \alpha^j > \alpha^{l'} - \frac{\alpha^{l'} - \alpha^l}{\alpha - 1} \\ &= \alpha^{l-1} \frac{\alpha^{l'-l}(\alpha^2 - 2\alpha) + \alpha}{\alpha - 1} \end{aligned}$$

Since  $\alpha \geq \frac{1+\sqrt{5}}{2}$  and  $l' - l \geq 1$ , we know that  $\frac{\alpha^{l'-l}(\alpha^2 - 2\alpha) + \alpha}{\alpha - 1} \geq \alpha(\alpha - 1) \geq 1$ . Therefore,  $d(p, q) \geq \alpha^{l-1}$ . The theorem proves.

The basic operation to build a cover tree is the insertion operation, as shown in Algorithm 1. Here  $Q_l$  is a subset of the objects at level  $l$  which may contain the new object  $p$  as a descendant. If  $S$  consists of  $n$  objects, the computational complexity of inserting a object is  $O(c^6 \log(n))$  [1]. A cover tree can be built on a data set  $S$  by sequentially calling Algorithm 1 to insert each object in  $S$ , with a total computational complexity of  $O(c^6 n \log(n))$ .

Although a cover tree is defined on levels from  $-\infty$  to  $\infty$ , the objects usually lie in a few levels. To measure the cover tree structure, we define the height of a cover tree as

$$H(CT) = tl(CT) - bl(CT) + 1 \tag{8}$$

where  $tl(CT)$  is the top level of  $CT$  which is defined as the highest level  $l$  such that  $|CS_l| \geq 1$ , i.e.,

$$tl(CT) = \max_l \{|CS_l| \geq 1\} \tag{9}$$

$bl(CT)$  is the bottom level of  $CT$  which is defined as the highest level  $l$  such that  $CR^l = 100\%$ , i.e.,

$$bl(CT) = \max_l \{|CS_l| = n\} \tag{10}$$

where  $n$  is the number of objects in  $S$ .

The height of a cover tree can be measured as follows.

**Theorem 2.** *The height of a cover tree  $CT$  built on  $S = \{s_1, \dots, s_n\}$  with metric  $d$  is bounded in  $H(CT) < \log_\alpha(\frac{d_{max}(S)}{d_{min}(S)}) + 3$ , where  $d_{max}(S) = \max_{s_i, s_j \in S} d(s_i, s_j)$  and  $d_{min}(S) = \min_{s_i, s_j \in S} d(s_i, s_j)$ .*

*Proof.* From the covering invariant of cover tree, we have  $d_{min}(S) < \alpha^{bl(CT)+1}$ , i.e.,  $bl(CT) > \log_\alpha(d_{min}(S)) - 1$ . From the separation invariant, we have  $d_{max}(S) > \alpha^{tl(CT)-1}$ , i.e.,  $tl(CT) < \log_\alpha d_{max}(S) + 1$ . Then  $H(CT) = tl(CT) - bl(CT) + 1 < \log_\alpha \frac{d_{max}(S)}{d_{min}(S)} + 3$ .

If a cover tree consists of too few or too many levels, more distance comparisons will be involved during the insertion and searching operations. If the data consists of  $n$  objects, we wish to build a balanced cover tree with high

approximate to  $\log_2 n$ . In order to build a balanced cover tree, we need to set the base parameter as  $\alpha \approx \max\{\frac{1+\sqrt{5}}{2}, \frac{d_{max}(S)}{d_{min}(S)}^{1/(\log_2^2 - 3)}\}$ .

To find the nearest neighbor of a point  $p$  from a cover tree  $CT$ , we descend through the tree level by level, keeping track of a subset  $Q_i \in CS_i$  of nodes that may contain the nearest neighbor of  $p$  as a descendant. The algorithm for finding the  $k$  nearest neighbors of an object  $p$  from a cover tree  $CT$  is shown in Algorithm 2, which iteratively constructs  $Q_{i-1}$  by expanding  $Q_i$  to its children in  $CS_{i-1}$  then throwing away any child  $q$  that cannot lead to the  $k$ -nearest neighborhoods of  $p$ . Note that although the algorithm is stated using an infinite loop over the implicit representation, it only needs to operate on the explicit representation (Fig. 1).

Figure 2 shows an example of 4-nearest neighbors search, in which nodes in light color represent the current found  $k$  nearest neighbors. Here,  $Q_i, Q, R$  and  $\beta$  are variables in Algorithm 2.

The following theorem ensures the correctness of Algorithm 2.

---

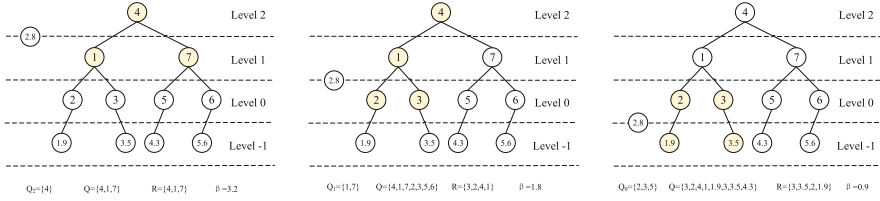
**Algorithm 2.** Find-kNN (Cover tree  $CT$ , object  $p$ , number of nearest neighbors  $k$ , base parameter  $\alpha$ )

---

- 1: **Input:** the nearest neighbor of  $p$  in  $CT$ .
  - 2: Set  $Q_\infty = C_\infty$ , where  $C_\infty$  is the root level of  $CT$ .
  - 3: Set  $R = C_\infty$
  - 4: **for**  $i = \infty$  to  $-\infty$  **do**
  - 5:     Set  $Q' = \{C_q : q \in Q_i\}$ .
  - 6:     Set  $Q = Q' \cup R$ .
  - 7:     **if**  $|Q| \geq k$  **then**
  - 8:         Form  $R$  with  $k$  nearest objects in  $Q$ , i.e.,  $|R| = \max k, |Q|$  and  $\forall q \in R$  and  $q' \in Q - R, d(p, q) \leq d(p, q')$ .
  - 9:          $\beta = \max_{q \in R} d(p, q)$ .
  - 10:        Form cover set  $Q_{i-1} = \{q \in Q' : d(p, q) \leq \beta + \frac{\alpha^i}{\alpha-1}\}$ .
  - 11:        **else**
  - 12:         Form cover set  $Q_{i-1} = \{q \in Q'\}$ .
  - 13:        **end if**
  - 14: **end for**
  - 15: If  $|Q| > k$ , update  $Q$  by only retaining the  $k$ -nearest neighbor of  $p$ . Return  $Q$  as the ultimate result.
- 

**Theorem 3.** Given a cover tree  $CT$  on  $\mathbf{D}$ , Find - kNN( $T, p, k$ ) returns the exact  $k$  nearest neighbors of  $p$  in  $\mathbf{D}$ .

*Proof.* For any  $q \in CS_{i-1}$ , the distance between  $q$  and any descendant  $q'$  is bounded by  $d(q, q') \leq \sum_{j=i-1}^{-\infty} \alpha^j = \frac{\alpha^i}{\alpha-1}$ . If  $q'$  is a  $k$  nearest neighbor of  $p$ , we have  $d(p, q') \leq \beta$ . Since  $d$  is a metric, according to the triangularity,  $d(p, q') \leq d(p, q) + d(q, q') \leq \frac{\alpha^i}{\alpha-1} + \beta$ . Therefore, step 10 can never throw out a grandparent of the  $k$  nearest neighbor of  $q$ . Eventually,  $k$  nearest neighbors are in  $Q_i$  and step 15 returns the exact  $k$  nearest neighbors of  $p$ .



(a) k-NN searching from  $L_2$  to (b) k-NN searching from  $L_1$  to (c) k-NN searching from  $L_0$  to  $L_{-1}$ .

**Fig. 1.** Process of finding 4 nearest neighbors in the cover tree from top to bottom by Algorithm 2.

### 5 Structured PurTree Spectral Clustering

Given a product tree  $\Psi$  and  $n$  purchase trees  $\Phi = \{\varphi_1, \dots, \varphi_n\}$ , we want to cluster  $\Phi$  into  $c$  clusters. Simultaneously, we want to learn the structured node weights  $\Omega$  to model the PurTree structure. Intuitively, the parent node is more important than its children nodes in distinguishing the different purchase trees. Therefore, we impose a constraint  $\sum_{t \in C_v(\Psi)} \omega_t = \omega_v$  on  $\Omega$  such the sum of children node weights equal to the parent node weights. The following theorem states the properties of  $\Omega$ :

**Theorem 4.**  $\forall 1 \leq l \leq H(\Phi), \sum_{v \in N^l(\Phi)} \omega_v = \frac{1}{H(\Phi)}$ .

*Proof.*  $\forall 1 \leq l \leq H(\Phi)$ , we can verify that

$$\sum_{v \in N^l(\Phi)} \omega_v = \sum_{v \in N^l(\Phi)} \sum_{t \in C_v(\Psi)} \omega_t = \sum_{v \in N^{l+1}(\Psi)} \omega_v$$

Then we have  $\sum_{v \in N^l(\Phi)} \omega_v = \frac{1}{H(\Phi)}$ .

Inspired by the work in [11], we wish to learn a new structured graph represented by an affinity matrix  $\mathbf{S} = [s_{ij}]_{n \times n}$ , in which  $s_{ij}$  is the probability that two trees  $\varphi_i$  and  $\varphi_j$  are connected. In order to find  $c$  clusters from  $\Phi$ , we hope that the graph constructed from  $\mathbf{S}$  only consists of  $c$  connected components. On the other side, we hope that  $\mathbf{S}$  best approximates the input affinity matrix. To achieve this goal, we present the following problem to simultaneously learn the node weights  $\Omega = \{\omega_v | v \in \cup_{l=1}^{H(\Phi)} N^l(\Phi)\}$  and the affinity matrix  $\mathbf{S}$

$$\begin{aligned} \min \quad & \left\| \mathbf{S} - \sum_{l=1}^{H(\Phi)} \sum_{v \in N^l(\varphi_i) \cup N^l(\varphi_j)} \omega_v \mathbf{A}^{(v)} \right\|_F^2 + \eta \sum_{l=1}^{H(\Phi)} \sum_{v \in N^l(\Phi)} \omega_v^2 \\ \text{s.t.} \quad & \mathbf{S}^T \mathbf{1} = \mathbf{1}, \mathbf{s} \succeq 0, \sum_{l=1}^{H(\Phi)} \sum_{v \in N^l(\Phi)} \omega_v = 1, \Omega \succeq 0, \sum_{t \in C_v(\Psi)} \omega_t = \omega_v, \text{rank}(\mathbf{L}_S) = n - c \end{aligned} \tag{11}$$

where  $\{\mathbf{A}^{(1)}, \dots, \mathbf{A}^{(\sum_{l=1}^{H(\Phi)} |N^l(\Phi)|)}\}$  is a set of node-based affinity matrices computed according to Eq. (7),  $\mathbf{L}_S = \mathbf{D}_S - \frac{\mathbf{S}^T + \mathbf{S}}{2}$  is the Laplacian matrix, the degree matrix  $\mathbf{D}_S \in \mathbb{R}^{n \times n}$  is defined as a diagonal matrix in which  $d_{ij} = \sum_{j=1}^n \frac{s_{ij} + s_{ji}}{2}$  and  $\eta$  is a regularization parameter.

According to the analysis in [10], problem (11) is equivalent to the following problem

$$\min \left\| \mathbf{S} - \sum_{l=1}^{H(\Phi)} \sum_{v \in N^l(\varphi_i) \cup N^l(\varphi_j)} \omega_v \mathbf{A}^{(v)} \right\|_F^2 + \eta \sum_{v \in N(\Phi)} \omega_v^2 + 2\mu Tr(\mathbf{F}^T \mathbf{L}_S \mathbf{F}) \quad (12)$$

where  $\mu$  is a large enough parameter.

We can apply the alternative optimization approach to solve problem (12).

### 5.1 Optimization of F

When  $\mathbf{S}$  and  $\Omega$  are fixed, problem (12) becomes

$$\min_{\mathbf{F} \in \mathbb{R}^{n \times c}, \mathbf{F}^T \mathbf{F} = \mathbf{I}} Tr(\mathbf{F}^T \mathbf{L}_S \mathbf{F}) \quad (13)$$

The optimal solution  $\mathbf{F}$  in problem (13) is formed by  $c$  eigenvectors of  $\mathbf{L}_S$  which corresponds to its  $c$  smallest eigenvalues.

### 5.2 Optimization of S

When  $\mathbf{F}$  and  $\Omega$  are fixed, problem (12) becomes

$$\min_{\mathbf{S}^T \mathbf{1} = \mathbf{1}, \mathbf{S} \geq 0} \left\| \mathbf{S} - \sum_{l=1}^{H(\Phi)} \sum_{v \in N^l(\Phi)} \omega_v \mathbf{A}^{(v)} \right\|_F^2 + \lambda \sum_{i,j=1}^n \|\mathbf{f}_i - \mathbf{f}_j\|_2^2 s_{ij} \quad (14)$$

where  $f_i \in \mathbb{R}^{c \times 1}$  is the transpose of the  $i$ -th row of  $\mathbf{F}$ .

Note that problem (14) is independent between different  $i$ , so we can solve it independently for each  $\mathbf{s}_i$  (the vector form of  $[s_{i1}, \dots, s_{in}]$ ) from the following problem

$$\min_{\mathbf{s}_i^T \mathbf{1} = 1, s_{ij} \geq 0} \sum_{j=1}^n (s_{ij} - a_{ij})^2 + \lambda \sum_{j=1}^n \|\mathbf{f}_i - \mathbf{f}_j\|_2^2 s_{ij} \quad (15)$$

Denote  $d_{ij}^f = \|\mathbf{f}_i - \mathbf{f}_j\|_2^2$  and  $\mathbf{d}_i^f \in \mathbb{R}^{n \times 1}$  as a vector with the  $j$ -th element as  $d_{ij}^f$ , problem (15) can be rewritten as

$$\min_{\mathbf{1}^T \mathbf{s}_i = 1, s_{ij} \geq 0} \left\| \mathbf{s}_i - \left( \mathbf{a}_i - \frac{\lambda \mathbf{d}_i^f}{2} \right) \right\|_2^2 \quad (16)$$

which has a closed-form solution and can be solved directly with the method in [14].

### 5.3 Optimization of $\Omega$

When  $\mathbf{S}$  and  $\mathbf{F}$  are fixed, problem (12) becomes

$$\min_{\sum_{l=1}^{H(\Phi)} \sum_{v \in N^l(\Phi)} \omega_v=1, \Omega \geq 0, \sum_{t \in C_v(\Phi)} \omega_t = \omega_v} \left\| \mathbf{S} - \sum_{v \in N(\varphi_i) \cup N(\varphi_j)} \omega_v \mathbf{A}^{(v)} \right\|_F^2 \quad (17)$$

According to Theorem 4, the sum of node weights in each level is a constant value  $\frac{1}{H(\Phi)}$ . Therefore, we can sequentially optimize the node weights level by level. We first let  $\omega_{Root} = \frac{1}{H(\Phi)}$ . Then given an internal node  $t \in N^l(\Phi)$ , we fix  $\omega_t$  and solve the following problem for  $\{\omega_v | v \in C_t\}$

$$\min_{\sum_{v \in C_t} \omega_v = \omega_t, \omega_v \geq 0} \left\| \mathbf{S}_V - \sum_{v \in C_t} \omega_v \mathbf{A}^{(v)} \right\|_F^2 \quad (18)$$

where  $\mathbf{S}_V = \mathbf{S} - \sum_{v \in N(\varphi_i) \cup N(\varphi_j)} \omega_v \mathbf{A}^{(v)} + \sum_{v \in C_t} \omega_v \mathbf{A}^{(v)}$ . Problem (18) can be rewritten as

$$\min_{\mathbf{1}^T \beta = \omega_t, \beta \geq 0} \beta^T \mathbf{G} \beta - \beta^T \mathbf{h} \quad (19)$$

where  $\beta \in \mathbb{R}^{|C_t| \times 1}$  consists of node weights of nodes in  $C_t$ .  $\mathbf{G}$  is a  $|C_t| \times |C_t|$  matrix in which  $g_{jl}$  is defined as

$$g_{jl} = Tr((\mathbf{A}^{(j)})^T \mathbf{A}^{(l)}) \quad (20)$$

and  $\mathbf{h}$  is a  $|C_t| \times 1$  column vector in which  $h_l$  is defined as

$$h_l = 2Tr(\mathbf{S}_V^T \mathbf{A}^{(l)}) \quad (21)$$

Problem (19) is difficult to directly solve since it involves  $\beta^T \mathbf{D} \beta$ . In this paper, we can use ALM to solve the above problem. Specifically, we need to solve the following problem

$$\min_{\beta^T \mathbf{1} = 1, \beta \geq 0, \gamma = \beta} \beta^T \mathbf{G} \gamma - \beta^T \mathbf{h} + \frac{\mu}{2} \left\| \beta - \gamma + \frac{\Lambda}{\mu} \right\|_F^2 \quad (22)$$

We use the alternating direction method of multipliers (ADMM) to solve this problem. Specifically, we optimize problem (22) with respect to one variable when fixing another variable, which results in the following two subproblems. When  $\gamma$  is fixed, problem (22) can be rewritten as

$$\min_{\beta^T \mathbf{1} = \omega_t, \beta \geq 0} \left\| \beta - \gamma + \frac{\Lambda + \mathbf{G} \gamma - \mathbf{h}}{\mu} \right\|_F^2 \quad (23)$$

When  $\beta$  is fixed, problem (22) can be rewritten as

$$\min_{\gamma^T \mathbf{1} = \omega_t, \gamma \geq 0} \left\| \gamma - \beta + \frac{\mathbf{G}^T \beta - \Lambda}{\mu} \right\|_F^2 \quad (24)$$

Problems (23) and (24) have closed-form solutions and can be solved directly with the method in [14].

### 5.4 The Optimization Algorithm

The detailed algorithm to solve problem (12), namely Structured PurTree Spectral Clustering (SPSS), is summarized in Algorithm 4. In this algorithm, we first initialize  $\Omega$  as follows

$$\omega_v = \begin{cases} \frac{1}{H(\Phi)} & \text{if } v = \text{Root}(\Psi) \\ \frac{\omega_t}{|C_t|} & \text{if } v \in C_t \end{cases} \quad (25)$$

and  $\mathbf{S} = \sum_{v \in N(\phi)} \omega_v \mathbf{A}^{(v)}$ . Then  $F$ ,  $\Omega$  and  $\mathbf{S}$  in (12) are iteratively solved until the algorithm converges. The clustering assignment is obtained from the graph constructed from  $\mathbf{S}$ .

**Algorithm 3.** Algorithm to solve problem (17)

- 1: **Input:** A set of node-based affinity matrices  $\mathbf{A}^{(1)}, \dots, \mathbf{A}^{(\sum_{l=1}^{H(\Phi)} |N^l(\Phi)|)}$ ,  $\mathbf{S}$ ,  $1 < \rho < 2$ .
- 2: Initialize  $\mu > 0$ ,  $\Lambda$ .
- 3: Set  $\omega_{\text{Root}} = \frac{1}{H(\Phi)}$ , initialize an empty stack  $Q$  and add root node into  $Q$ .
- 4: **repeat**
- 5:     Pop a node  $t$  from  $Q$ .
- 6:     **repeat**
- 7:         Update  $\beta_{t+1}$  by solving problem (23).
- 8:         Update  $\gamma_{t+1}$  by solving problem (24).
- 9:         Update  $\Lambda$ , which  $\Lambda = \Lambda + \mu(\beta_{t+1} - \gamma_{t+1})$ .
- 10:         Update  $\mu$ , which  $\mu = \rho\mu$ .
- 11:     **until** problem (22) converges
- 12:     Assign  $\beta$  to  $\{\omega_v | v \in C_t\}$ .
- 13:     **for** Each node  $s \in C_t$  **do**
- 14:         Add  $s$  into  $Q$  if  $s$  is an internal node.
- 15:     **end for**
- 16: **until**  $Q$  is empty
- 17: **Output:**  $\Omega$ .

**Algorithm 4.** Structured PurTree Spectral Clustering algorithm to solve problem (12)

- 1: **Input:** A set of  $n$  purchase trees  $\Phi = \{\varphi_1, \dots, \varphi_n\}$ , the number of nearest neighbors  $k$ , the number of clusters  $c$ .
- 2: Compute the similarity matrices  $\{\mathbf{A}^{(1)}, \dots, \mathbf{A}^{(|N(\Phi)|)}\}$  according to Eq. (7), where Algorithm 2 is used to find the  $k$ -nearest neighborhoods of any object.
- 3: Initialize  $\Omega$  according to Eq. (25) and randomly initialize  $\mathbf{S}$ .
- 4: **repeat**
- 5:     Update  $\mathbf{F}$  by selecting  $c$  eigenvector of  $\mathbf{L}_S = \mathbf{D}_S - \frac{\mathbf{S}^T + \mathbf{S}}{2}$  which corresponds to its  $c$  smallest eigenvalues.
- 6:     Update  $\mathbf{S}$  by solving problem (16).
- 7:     Update  $\Omega$  by calling Algorithm 3.
- 8: **until** (converges)
- 9: **Output:** Find the connected components in the graph constructed from  $\mathbf{S}$ , and assign the objects in the same connected component to the same cluster.

Since each of three variables  $\mathbf{F}$ ,  $\mathbf{S}$  and  $\mathbf{\Omega}$  are alternatively optimized, Algorithm 4 will decrease problem (12) in each iteration. Note that although Algorithm 3 does not monotonically decrease problem (17), the converged  $\mathbf{\Omega}$  will decrease problem (17). Therefore, the convergency of Algorithm 4 is ensured. If the algorithm needs  $r$  iterations to converge, the complexity of SPSS algorithm is  $O(r(kn^2 + N(\Phi)))$ , where the eigendecomposition of  $\mathbf{L}_S$  takes  $O(kn^2)$  time because  $\mathbf{S}$  is a  $k$ -NN affinity matrix.

## 6 Experimental Results and Analysis

10 real-world transaction data sets, shown in Table 1, were used to investigate the new proposed method.  $D_1$  was built from a superstore’s transactional data set<sup>1</sup>, which consists of 8,399 transaction records from 796 customers.  $D_2 \sim D_6$  were built from five subsets of a super market’s transactional data, which contains up to 25 million newest transaction records.  $D_7 \sim D_{10}$  were built from four subsets of one year’s purchase history transaction data from the kaggle competition<sup>2</sup>, which contains more than 349 million transaction records.

**Table 1.** Characteristics of 10 customer transaction data sets.

Data sets ( $D$ )	Size	$ N^2(D) $	$ N^3(D) $	$ N^4(D) $	$ N^5(D) $
$D_1$	795	3	17	1264	
$D_2$	208	84	501	1198	9760
$D_3$	416	90	552	1339	14274
$D_4$	832	90	606	1457	18137
$D_5$	1665	91	651	1584	22822
$D_6$	3330	91	697	1714	28065
$D_7$	608	821	73164	89665	
$D_8$	1216	826	74500	91785	
$D_9$	2433	825	75737	93731	
$D_{10}$	4867	827	77424	96538	

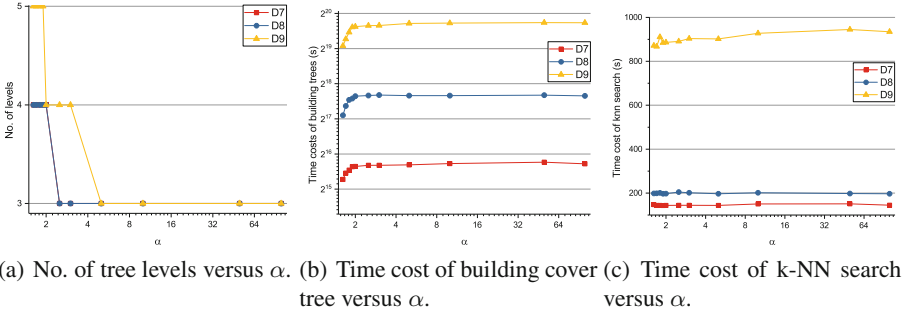
Since the 10 data sets in Table 1 contain no labels, we use the normalized  $\log(W_k)$  to evaluate the clustering results [4], which is computed as

$$\mathcal{NLW}(\mathcal{C}) = \log\left\{\sum_{l=1}^c \frac{1}{2|C_l|} \sum_{i,j \in C_l} d(\varphi_i, \varphi_j)\right\} - \log\left(\sum_{i,j=1}^n d(\varphi_i, \varphi_j)\right) \quad (26)$$

where  $\mathcal{C}$  consists of  $c$  clusters. The lower the  $\mathcal{NLW}(\mathcal{C})$ , the better the clustering result. To compute  $\mathcal{NLW}(\mathcal{C})$  for SPSS, we compute the distance  $d(\varphi_i, \varphi_j) = 1 - a_{ij}$  where  $a_{ij}$  is the PurTree subspace similarity between  $\varphi_i$  and  $\varphi_j$ .

<sup>1</sup> <https://community.tableau.com/docs/DOC-1236>.

<sup>2</sup> <http://www.kaggle.com/c/acquire-valued-shoppers-challenge/data>.



**Fig. 2.** Experimental results of k-NN search on  $D_7$ ,  $D_8$  and  $D_9$ .

### 6.1 Experimental Results on $k$ -NN Search

In this experiment, we analyse the impact of  $\alpha$  to Algorithm 2. We set  $\alpha$  to  $\{1.62, 1.7, 1.8, 2.0, 2.5, 3, 5, 10, 50, 100\}$  to build a set of cover trees based on the distance of level  $L_1$  on  $D_7$ ,  $D_8$  and  $D_9$ . Then we performed k-NN search with these built cover trees, by setting the number of nearest neighbors  $k$  to  $\{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$ . The experimental results are drawn in Fig. 2. Figure 2(a) indicates that the number of tree levels decreases as  $\alpha$  increases, as a result of rapid increasing of the time cost of building cover trees as shown in Fig. 2(b). Figure 2(c) indicates that the time cost of k-NN search does not change too much as  $\alpha$  increases. This may be because that the number of tree levels does not change too much in the three datasets. In real applications, we can simply set  $\alpha$  to 1.62.

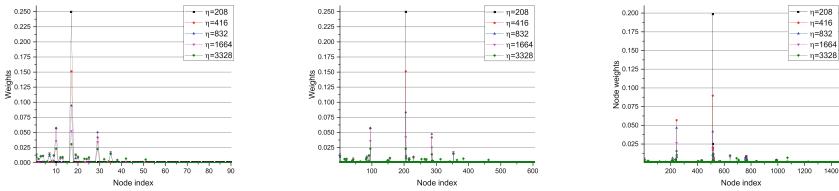
**Table 2.** Comparison of  $\mathcal{NLW}$  (Mean  $\pm$  Standard Deviation) of seven clustering algorithms on 10 benchmark data sets (The best result on each data set is highlighted in bold).

Data	DBSCAN	HAC	NCut	RCut	PurTreeClust	LPS	SPSS
$D_1$	-7.47 $\pm$ 1.23	-7.75 $\pm$ 1.24	-7.41 $\pm$ 0.24	-7.41 $\pm$ 0.24	-7.83 $\pm$ 0.80	-7.60 $\pm$ 0.36	<b>-9.90</b> $\pm$ 0.80
$D_2$	-6.03 $\pm$ 0.04	-6.34 $\pm$ 0.90	-6.17 $\pm$ 0.40	-6.19 $\pm$ 0.40	-6.31 $\pm$ 0.80	-6.32 $\pm$ 0.35	<b>-6.58</b> $\pm$ 0.21
$D_3$	-6.73 $\pm$ 0.04	-6.94 $\pm$ 0.64	-6.81 $\pm$ 0.20	-6.81 $\pm$ 0.20	-6.91 $\pm$ 0.64	-7.02 $\pm$ 0.32	<b>-7.12</b> $\pm$ 0.21
$D_4$	-7.42 $\pm$ 0.04	-7.59 $\pm$ 0.63	-7.48 $\pm$ 0.18	-7.48 $\pm$ 0.24	-7.62 $\pm$ 0.52	-7.58 $\pm$ 0.36	<b>-7.72</b> $\pm$ 0.2
$D_5$	-8.11 $\pm$ 0.08	-8.26 $\pm$ 0.45	-8.16 $\pm$ 0.15	-8.15 $\pm$ 0.15	-8.34 $\pm$ 0.38	-8.38 $\pm$ 0.33	<b>-8.88</b> $\pm$ 0.25
$D_6$	-8.80 $\pm$ 0.08	-8.92 $\pm$ 0.58	-8.84 $\pm$ 0.12	-8.84 $\pm$ 0.12	-8.89 $\pm$ 0.41	-9.02 $\pm$ 0.34	<b>-9.76</b> $\pm$ 0.34
$D_7$	-7.10 $\pm$ 0.08	-7.19 $\pm$ 0.16	-7.17 $\pm$ 0.12	-7.16 $\pm$ 0.12	-7.40 $\pm$ 0.49	-7.62 $\pm$ 0.35	<b>-8.12</b> $\pm$ 0.46
$D_8$	-7.80 $\pm$ 0.08	-7.85 $\pm$ 0.14	-7.83 $\pm$ 0.12	-7.83 $\pm$ 0.12	-7.90 $\pm$ 0.47	-8.08 $\pm$ 0.34	<b>-8.79</b> $\pm$ 0.49
$D_9$	-8.49 $\pm$ 0.08	-8.53 $\pm$ 0.13	-8.52 $\pm$ 0.12	-8.51 $\pm$ 0.12	-8.58 $\pm$ 0.49	-8.58 $\pm$ 0.38	<b>-9.37</b> $\pm$ 0.48
$D_{10}$	-9.18 $\pm$ 0.08	-9.21 $\pm$ 0.12	-9.20 $\pm$ 0.08	-9.20 $\pm$ 0.08	-9.30 $\pm$ 0.50	-9.4 $\pm$ 0.35	<b>-9.92</b> $\pm$ 0.45

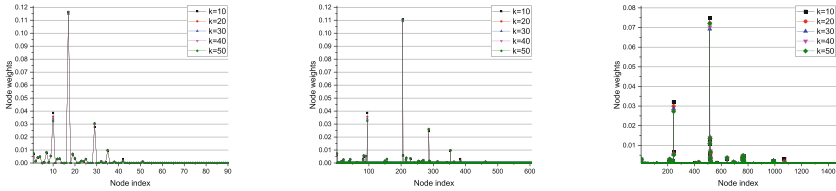


### 6.2 Results and Analysis

We used all ten data sets to compare the effectiveness of the SPSS algorithm with six clustering algorithms, i.e., DBSCAN, HAC, Ncut [8], RCut [5] and PurTreeClust [2] and LPS [4]. In this experiment,  $\alpha$  was set to 1.62 and we selected 96 integers from 5 to 100 for  $c$ . The PurTree subspace similarity is used for DBSCAN, HAC, Ncut, RCut and SPSS, and the PurTree distance is used for PurTreeClust in which  $\gamma$  was set as the same values used in [2], i.e.,  $\gamma = \{0, 0.2, 0.8, 1, 2, 1000\}$ . 20 integers from 5 to 100 were used for  $k$  in DBSCAN, PurTreeClust and SPSS. On each data set,  $\eta$  in SPSS was set to  $\{\frac{1}{4}, \frac{1}{2}, 1, 2, 3\}$  times of the number of objects in the data set. The other parameters of all methods were set with the same strategy to make fair comparison, i.e.,  $\{10^{-3}, 10^{-2}, \dots, 10^3\}$ . For DBSCAN, we also set eps as 95 values from 0.25 to 0.5 and minPts as 95 values from 1% to 10% of the number of objects. For each clustering algorithm, we computed the average  $\mathcal{NLW}$  and show the results in Table 2 which show that SPSS outperformed all the other methods on all sets. For example, on  $D_7, D_8, D_9$  and  $D_{10}$  which consist of more than 80 thousands of products, SPSS achieves a greater than 7% improvement compared to the second-best method LPS. On  $D_1$ , SPSS even achieves a nearly 30% improvement compared to the second-best method PurTreeClust. Besides SPSS, LPS produced better results than the other methods.



(a)  $\{\omega_v | v \in N^2(\Phi)\}$  versus  $\eta$ . (b)  $\{\omega_v | v \in N^3(\Phi)\}$  versus  $\eta$ . (c)  $\{\omega_v | v \in N^4(\Phi)\}$  versus  $\eta$ .



(d)  $\{\omega_v | v \in N^2(\Phi)\}$  versus  $k$ . (e)  $\{\omega_v | v \in N^3(\Phi)\}$  versus  $k$ . (f)  $\{\omega_v | v \in N^4(\Phi)\}$  versus  $k$ .

**Fig. 3.** Node weights  $\Omega = \{\omega_v\}$  versus  $\eta$  and  $k$  on  $D_4$ .

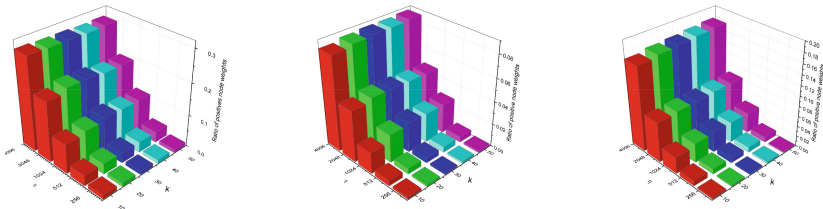
### 6.3 Parameter Sensitivity Study

Figures 3(a), (b) and (c) show the weights of the second, third and fourth levels of nodes versus  $\eta$ , and Figs. 3(d), (e) and (f) show the weights of the second, third and fourth levels of nodes versus  $k$ , respectively.

For each  $\eta$  and  $k$ , we computed the average node weights on  $D_4$  and draw them in Fig. 3. These figures show that the node weights are nearly stable as  $\eta$  and  $k$  increase. We also observe that the node weight distributions in different levels are highly correlated. Specifically, the average weight decreases level by level. This can be verified according to Theorem 4 which indicates that the sum of weights for nodes in different levels are equal, and the fact that the high level contains more nodes than low level.

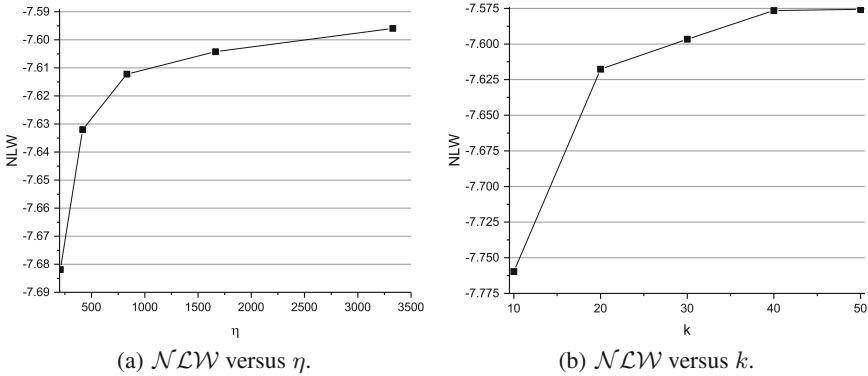
We compute the ratio of positive weights versus  $\eta$  and  $k$ , and draw the results in Fig. 4. From these figures, we can see that the number of positive weights is sensitive to  $\eta$ , but insensitive to  $k$ . With the increase of the tree level, the number of positive weights decreases rapidly.

Finally, we show the relationship between  $\mathcal{NLW}$  and the two parameters  $\eta$  and  $k$  on  $D_4$  in Fig. 5. From this figure, we can observe that  $\mathcal{NLW}$  increases as



(a) Ratio of positive weights for nodes in  $L_2$ . (b) Ratio of positive weights for nodes in  $L_3$ . (c) Ratio of positive weights for nodes in  $L_4$ .

**Fig. 4.** Ratio of positive weights versus  $\eta$  and  $k$ .



(a)  $\mathcal{NLW}$  versus  $\eta$ . (b)  $\mathcal{NLW}$  versus  $k$ .

**Fig. 5.**  $\mathcal{NLW}$  versus  $\eta$  and  $k$  on  $D_4$ .

both  $\eta$  and  $k$  increase. In real applications, we can perform hierarchy grid search to select the proper  $m$  and  $k$  for better results.

## 7 Conclusion

We have presented SPSS, a Structured PurTree Subspace Spectral clustering algorithm for customer transaction data. In the new method, a PurTree subspace similarity is proposed to compute the similarity between two purchase trees. A new clustering model is proposed to learn a structured graph that best approximating the input similarity, and a set of sparse and structured node weights are learned to better recover the cluster structure. An iterative optimization algorithm is proposed to simultaneously learn the graph and node weights. Experimental results on 10 real-world data sets have demonstrated the superior performance of the new method. In future work, we will improve our method for transaction data which contains a large number of customers.

**Acknowledgment.** This research was supported by the National Key R&D Program of China 2018YFB1003201, NSFC under Grant no. 61773268, 61502177 and U1636202, and Guangdong Key Laboratory project 2017B030314073.

## References

1. Beygelzimer, A., Kakade, S., Langford, J.: Cover trees for nearest neighbor. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 97–104. ACM (2006)
2. Chen, X., Fang, Y., Yang, M., Nie, F., Zhao, Z., Huang, J.Z.: PurTreeClust: a clustering algorithm for customer segmentation from massive customer transaction data. *IEEE Trans. Knowl. Data Eng.* **30**(3), 559–572 (2018). <https://doi.org/10.1109/TKDE.2017.2763620>
3. Chen, X., Huang, J.Z., Luo, J.: PurTreeClust: a purchase tree clustering algorithm for large-scale customer transaction data. In: 2016 IEEE 32nd International Conference on Data Engineering (ICDE), pp. 661–672, May 2016. <https://doi.org/10.1109/ICDE.2016.7498279>
4. Chen, X., Peng, S., Huang, J.Z., Nie, F., Ming, Y.: Local PurTree spectral clustering for massive customer transaction data. *IEEE Intell. Syst.* **32**(2), 37–44 (2017)
5. Hagen, L., Kahng, A.B.: New spectral methods for ratio cut partitioning and clustering. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **11**(9), 1074–1085 (1992)
6. Kuo, R., Ho, L., Hu, C.M.: Integration of self-organizing feature map and k-means algorithm for market segmentation. *Comput. Oper. Res.* **29**(11), 1475–1493 (2002)
7. Lu, T.C., Wu, K.Y.: A transaction pattern analysis system based on neural network. *Expert Syst. Appl.* **36**(3), 6091–6099 (2009)
8. Ng, A.Y., Jordan, M.I., Weiss, Y., et al.: On spectral clustering: analysis and an algorithm. In: *Advances in Neural Information Processing Systems 2*, pp. 849–856 (2002)
9. Ngai, E.W., Xiu, L., Chau, D.C.: Application of data mining techniques in customer relationship management: a literature review and classification. *Expert Syst. Appl.* **36**(2), 2592–2602 (2009)

10. Nie, F., Wang, X., Huang, H.: Clustering and projected clustering with adaptive neighbors. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 977–986. ACM (2014)
11. Nie, F., Wang, X., Jordan, M.I., Huang, H.: The constrained Laplacian rank algorithm for graph-based clustering. In: Thirtieth AAAI Conference on Artificial Intelligence, pp. 1969–1976 (2016)
12. Tsai, C.Y., Chiu, C.C.: A purchase-based market segmentation methodology. *Expert Syst. Appl.* **27**(2), 265–276 (2004)
13. Wang, K., Xu, C., Liu, B.: Clustering transactions using large items. In: Proceedings of the Eighth International Conference on Information and Knowledge Management, pp. 483–490. ACM (1999)
14. Wang, W., Carreira-Perpián, M.Á.: Projection onto the probability simplex: an efficient algorithm with a simple proof, and an application. *Mathematics* (2013)
15. Xiao, Y., Dunham, M.H.: Interactive clustering for transaction data. In: Kambayashi, Y., Winiwarter, W., Arikawa, M. (eds.) DaWaK 2001. LNCS, vol. 2114, pp. 121–130. Springer, Heidelberg (2001). [https://doi.org/10.1007/3-540-44801-2\\_13](https://doi.org/10.1007/3-540-44801-2_13)
16. Xiong, T., Wang, S., Mayers, A., Monga, E.: DHCC: Divisive hierarchical clustering of categorical data. *Data Mining Knowl. Discovery* **24**(1), 103–135 (2012)



# Dynamic Stochastic Block Model with Scale-Free Characteristic for Temporal Complex Networks

Xunxun Wu<sup>1</sup>, Pengfei Jiao<sup>2</sup>(✉), Yaping Wang<sup>1</sup>, Tianpeng Li<sup>1</sup>, Wenjun Wang<sup>1</sup>,  
and Bo Wang<sup>1</sup>

<sup>1</sup> College of Intelligence and Computing, Tianjin University, Tianjin, China  
{xxwu, yapingwang, ltpnimeia, wjwang, bo\_wang}@tju.edu.cn

<sup>2</sup> Center for Biosafety Research and Strategy, Tianjin University, Tianjin, China  
pjiao@tju.edu.cn

**Abstract.** Complex network analysis has been widely applied in various fields such as social system, information system, and biological system. As the most popular model for analyzing complex network, Stochastic Block Model can perform network reconstruction, community detection, link prediction, anomaly detection, and other tasks. However, for the dynamic complex networks which are always modeling as a series of snapshot networks, the existing works for dynamic networks analysis which are based on the stochastic block model always analyze the evolution of dynamic networks by introducing probability transition matrix, then, the scale-free characteristic (power law of the degree distribution) of the network, is ignoring. So in order to overcome this limitation, we propose a fully Bayesian generation model, which incorporates the heterogeneity of the degree of nodes to model dynamic complex networks. Then we present a new dynamic stochastic block model for community detection and evolution tracking under a unified framework. We also propose an effective variational inference algorithm to solve the proposed model. The model is tested on the simulated datasets and the real-world datasets, and the experimental results show that the performance of it is superior to the baselines of community detection in dynamic networks.

**Keywords:** Dynamic community detection ·  
Scale-free characteristics · Stochastic block models · Bayesian inference

## 1 Introduction

With the advent of the big data era, network data has been widely developed in various fields such as human social interaction, academic cooperation, and biological information, and occupies an increasing role. How to mine useful information from the network data has become the focus of our attention. Analysis and modeling of the complex network can help us better understand the structure and properties and mining important information in network data.

Community detection is an important research direction in complex networks analysis, which has important applications in various networks, such as social and biological networks. For example, in social networks, individuals with more active and frequent connections tend to have more similar interests and preferences [16]. In a citation network, the authors with more closer cooperation are likely belonging to the same research direction, which can help us to analyze the future research direction. In criminal networks, detecting potential crime gangs or terrorist organizations plays an important role in national security.

Due to the wide applications of community detection, lots of methods have been proposed, such as modularity-based method, clique-based method, spectral method and so on [20]. However, these studies ignore that the real network is dynamic, and the community structure will change over time. For example, in a criminal network, the members of criminal gangs are likely to change after a criminal act. In a citation network, researchers may also change their research direction due to the change of personal research interest or the influence of influential authors. The above static methods cannot capture the dynamic changes of the community and are not applicable to the temporal complex network.

In fact, the changes of nodes and structures of the dynamic network bring many challenges. Recently, many researchers turn their attention to the field of dynamic network community detection and put forward a series of dynamic community detection methods, such as heuristic methods, two-stage methods, evolutionary clustering, incremental clustering and so on.

Among all the methods, whether for dynamic or static networks, the most important one is the statistical network model. The statistical models deal with uncertainty through statistical inference and are an important tool in community detection. Stochastic Block Model (SBM) [6] plays an important role in both static and dynamic community detection [5,9]. It is a universal and popular method with strong theoretical explanatory power, has received great attention from researchers, and has been developed with various extensions [1,8]. The assumption of SBM is that the nodes in the network are divided into different communities, and the existence of the connection between pairwise nodes depends only on the communities the nodes belonging to. As a result of this assumption, the nodes within the community are considered to be indistinguishable, and the heterogeneity of node degree in the network is not considered. Therefore, the SBM cannot depict the scale-free characteristics in the real network, so it will cause significant bias when it is applied to the community detection. Considering this limitation, many degree retention methods in the static network are proposed, some of which use the degree retention within the community and only consider local information [11], other methods directly make use of the degree of nodes, so it is necessary to know the degree of nodes in advance, which leads to a consequence that the proposed method is not a complete generation model. However, to our best knowledge, there is no work proposed based on SBM to model the heterogeneity of nodes for community detection in dynamic networks.

Considering the limitations of existing community detection methods, we propose a dynamic network community detection model, called DPSBM, which combine with the popularity of nodes to detect and track the community structure in the dynamic network. The main contribution of our work includes

1. As we know, we are the first to consider the heterogeneity of node degree to model the generation process of dynamic networks for community detection.
2. The proposed DPSBM can simultaneously solve community detection, evolution tracking and the popularity of nodes simultaneously.
3. We propose an effective variational EM algorithm to optimize the objective function, and the experiments show that our model has better performance compared with baselines on community detection and evolution.

## 2 Related Work

Here we introduce some widely used methods for community detection in dynamic networks, the development of the SBM and the recent works of dynamic networks analysis based on dynamic Stochastic Block Model.

**Community Detection in Dynamic Networks:** In the traditional two-stage method [7], community detection and evolution tracking are two separate stages. Because real-world data is often noisy, the two-step approach can lead to unstable community structures and unexplained community evolution. Lee et al. [12] proposed an incremental tracking framework for cluster evolution over highly dynamic networks. Lin et al. [13] proposed FacetNet, on the basis of non-negative matrix decomposition, communities are regularized by the temporal smoothness of evolutions so that the community detection results not only conform to the current snapshot structure but also remain stable with the community detection of previous network snapshot. Folino et al. [2] proposed a detection method based on evolutionary clustering, DYNMOGA, which builds a multiobjective problem based on genetic algorithms and controls the preference degree of a user towards either the snapshot quality or the temporal quality by controlling an input parameter, so as to optimize the two competing objectives.

**Stochastic Block Model:** It was initially proposed by Holland et al. [6], and has been well applied in various fields such as social network and biological information. But SBM doesn't consider node's heterogeneity. Some scholars have noticed the problem of node heterogeneity, which has been continuously extended on the basis of SBM. Karrer and Newman et al. [11] considered the degree change of nodes and proposed a community detection method based on degree correction. However, only the degree of nodes within the community is considered, while the degree information on the whole network level is not considered, so this method is not comprehensive.

**Dynamic Stochastic Block Model:** Yang et al. [19] first proposed the Dynamic Stochastic Block Model (DSBM) based on SBM, built the probabilistic transfer matrix of nodes on the stochastic block model to simulate the transfer

probability of nodes between communities, and detected community and tracked community evolution, at the same time, proposed a combined solution algorithm based on Gibbs sampling and simulated annealing algorithm. Tang et al. [15] proposed the Dynamic Stochastic Blockmodel with Temporal Dirichlet Process (DBTDP), which extends the Stochastic Blockmodel by introducing Dirichlet processes and Chinese restaurant processes, in addition, this model enables the number of communities to be determined automatically at each time snapshot. These methods focus on the evolution of community structures in dynamic networks, but ignore the heterogeneity of nodes degree and cannot deal with the scale-free properties of networks in the real world.

### 3 Method

Before discussing the proposed model, we first introduce the notations. We use  $W = \{W^{(1)}, \dots, W^{(T)}\}$  to denote the set of snapshot network for a given network over  $T$  discrete time steps, where each element of  $W^{(t)}$  is the connection weights between nodes in time step  $t$ , and  $n$  is the number of nodes in the snapshot of the dynamic network. Without loss of generality, we consider an undirected and unweighted network, which can also be easily extended to a directional weighted network. If  $w_{ij}^{(t)} = 1$ , there exists a connection between node  $i$  and node  $j$ , if  $w_{ij}^{(t)} = 0$ , there is not a connection between node  $i$  and node  $j$ .

We use  $Z = \{Z^{(1)}, \dots, Z^{(T)}\}$  to denote the community ownership of all nodes in all snapshots. In each time step, we use  $z_i \in \{1, \dots, K\}$  to denote the community ownership of node  $i$ , where  $K$  is the number of communities in the network. In other words,  $z_i = k$  means that node  $i$  belongs to the  $k$ th community.

#### 3.1 SBM

We start with a brief introduction to SBM. SBM is a classic stochastic block model suitable for static networks, which has been successfully applied in various fields. The generation mechanism of the SBM model obeys the following rules. First, we assign each node in the network to the community by the multinomial distribution with the parameter of  $\pi$ , where  $\pi = \{\pi_1, \dots, \pi_K\}$ . Then, the generation of the connection between node  $i$  and node  $j$  follows a Bernoulli distribution with a parameter of  $b_{kl}$  (here let  $z_i = k$ ,  $z_j = l$ ), where  $B \in [0, 1]^{K \times K}$  is the block matrix, where  $b_{kl}$  is the probability of the connection between the node in the community  $k$  and the node in the community  $l$ . When  $k = l$ ,  $b_{kl}$  is the probability of connection between nodes within the community. When  $k \neq l$ ,  $b_{kl}$  is the probability of connection between nodes belonging to different communities. This is an important assumption of the SBM, that the generation of links between nodes is only related to the community of nodes.

#### 3.2 DPSBM

Because the traditional stochastic block model is only applicable to the static network, it does not conform to the dynamic evolution of networks. At the same time, SBM regards the nodes within the community as undifferentiated





2. The community assignment  $z_i^{(t)}$  of node  $i$  in time step  $t$  is independent of other nodes and links, and depends on the community assignment  $z_i^{(t-1)}$  in time step  $t - 1$ .
3. The popularity  $\delta_i^{(t)}$  of node  $i$  in time step  $t$  is independent of other nodes, and depends on the popularity  $\delta_i^{(t-1)}$  of node  $i$  in time step  $t - 1$ .

**Table 1.** Notations

Type	Symbol	Description
scalars	$n$	number of nodes
	$K$	number of communities
	$T$	number of time steps
observation variable	$w_{ij}^{(t)}$	connection between nodes $i$ and $j$ in time step $t$
latent variables	$z_i^{(t)}$	community that node $i$ belongs to in time step $t$
	$\delta_i^{(t)}$	the popularity of node $i$ in time step $t$
parameter variables	$b_{kl}$	probabilities of connection between communities $k$ and $l$
	$A_{kl}$	transfer probabilities of node in adjacent time step
	$\pi_k$	probability that a node is assigned to community $k$
hyperparameters	$\lambda$	the parameter of the distribution of node popularity
	$\Sigma$	the variance of Gauss Process

According to the above three assumptions, the generative process of edges is summarized as follows, where the graphical model of DPSBM is shown in Fig. 1, and the notation representation is illustrated in Table 1 (including five types of scalars, observation variables, latent variables, parameter variables and hyperparameters).

1. For time step  $t = 1$ :
  - (a) For each node  $i \in \mathcal{N} = \{1, 2, \dots, N\}$ :
    - i. sample the cluster index  $z_i^{(1)} \sim \text{Multi}(\pi)$ ;
    - ii. sample the popularity  $\delta_i^{(1)} \sim \text{Exp}(\lambda)$ ;
  - (b) For each node-pair  $(i, j) \in \mathcal{N} \times \mathcal{N}$ :
    - i. sample the link  $w_{ij}^{(1)} \sim \text{Bern}(b_{z_i^{(1)} z_j^{(1)}}^{1+\delta_i^{(1)}+\delta_j^{(1)}})$ ;

2. For each time step  $t > 1$ :
  - (a) For each node  $i \in \mathcal{N} = \{1, 2, \dots, N\}$ :
    - i. sample the cluster index  $z_i^{(t)} \sim p(z_i^{(t)} | z_i^{(t-1)}, A)$ ;
    - ii. sample the popularity  $\delta_i^{(t)} \sim \mathbf{N}(\delta_i^{(t-1)}, \Sigma)$ ;
  - (b) For each node-pair  $(i, j) \in \mathcal{N} \times \mathcal{N}$ :
    - i. sample the link  $w_{ij}^{(t)} \sim \text{Bern}(b_{z_i^{(t)} z_j^{(t)}}^{1+\delta_i^{(t)}+\delta_j^{(t)}})$

The joint probability distribution of the observable variables  $W$  and latent variables  $Z$  and  $\delta$  in our model is as follows:

$$\begin{aligned}
 &Pr(W, Z, \delta | \pi, B, A, \lambda) \\
 &= \prod_{t=1}^T Pr(W^{(t)} | Z^{(t)}, B, \delta^{(t)}) \prod_{t=2}^T Pr(Z^{(t)} | Z^{(t-1)}, A) \\
 &Pr(Z^{(1)} | \pi) Pr(\delta^{(1)} | \lambda) \prod_{t=2}^T Pr(\delta^{(t)} | \delta^{(t-1)}, \Sigma)
 \end{aligned} \tag{1}$$

### 4 Inference

In this section, we propose a variational EM algorithm to infer DPSBM.

We use the lower bound (ELBO)  $\mathcal{L}(Z, \delta; \pi, B, A, \lambda)$  to approximate the log marginal probability  $\log Pr(W | \pi, B, A, \lambda)$ , and we use the mean-field method to decompose  $q(Z, \delta)$  into:

$$q(Z, \delta) = \prod_t \left[ \prod_i q(z_i^{(t)}) \prod_i q(\delta_i^{(t)}) \right] \tag{2}$$

where  $q(z_i^{(t)}) = \text{Multi}(\phi_i^{(t)})$ , and  $q(\delta_i^{(t)}) = \mathbf{1}(\bar{\delta}_i^{(t)})$ .  $q(z_i^{(t)})$  is a multinomial distribution with the parameter  $\phi_i^{(t)}$ , and  $q(\delta_i^{(t)})$  is a degenerated distribution with the parameter  $\bar{\delta}_i^{(t)}$ .

Since most of the networks we focus on are sparse and scale-free, in other words,  $b_{kl}$  and most of  $\delta_i$  are not large. By using Taylor’s approximation, the ELBO is given by:

$$\begin{aligned}
 &\mathcal{L}(Z, \delta; \pi, B, A, \lambda) \\
 &= E_q \log Pr(W, Z, \delta | \pi, B, A, \lambda) - E_q \log q(Z, \delta) \\
 &\approx \sum_{t=1}^T \sum_{w_{ij}^{(t)}=1} (1 + \bar{\delta}_i^{(t)} + \bar{\delta}_j^{(t)}) \sum_k \sum_l \phi_{ik}^{(t)} \phi_{jl}^{(t)} \log b_{kl} - \sum_{t=1}^T \sum_{w_{ij}^{(t)}=0} \sum_k \sum_l \phi_{ik}^{(t)} \phi_{jl}^{(t)} b_{kl}^{1+\bar{\delta}_i^{(t)}+\bar{\delta}_j^{(t)}} \\
 &+ \sum_{t=2}^T \sum_i \sum_l \sum_k \phi_{il}^{(t-1)} \phi_{ik}^{(t)} \log A_{lk} + \sum_i \sum_k \phi_{ik}^{(1)} \log \pi_k + N \log \lambda - \lambda \sum_i \bar{\delta}_i^{(1)} \\
 &- \sum_{t=2}^T \left[ \frac{1}{2} N \log 2\pi \Sigma + \frac{1}{2\Sigma} \sum_i (\bar{\delta}_i^{(t)} - \bar{\delta}_i^{(t-1)})^2 \right] - E_q \log q
 \end{aligned} \tag{3}$$

In the following two sections, we maximize the lower bound (ELBO) to obtain the variational parameters  $(\phi, \delta)$  and model parameters  $(\pi, B, A)$ .

#### 4.1 E-Step

We fixed the model parameters  $(\pi, B, A)$  and maximized ELBO to obtain the variational parameters  $(\phi, \delta)$ .

For  $t = 1$ , the updating equation for optimal  $\phi_{ik}^{(1)}$  is given by,

$$\begin{aligned} \phi_{ik}^{(1)} \propto \pi_k \exp\{ & \sum_{w_{ij}^{(1)}=1} (1 + \bar{\delta}_i^{(1)} + \bar{\delta}_j^{(1)}) \sum_l \phi_{jl}^{(1)} \log b_{kl} \\ & - \sum_{w_{ij}^{(1)}=0} \sum_l \phi_{jl}^{(1)} b_{kl}^{1+\bar{\delta}_i^{(1)}+\bar{\delta}_j^{(1)}} + \sum_l \phi_{il}^{(2)} \log A_{kl} \} \end{aligned} \quad (4)$$

For  $\bar{\delta}_i^{(1)}$ , we use gradient ascend to obtain its optimal solution. The gradient is derived as:

$$\begin{aligned} \frac{\partial \mathcal{O}(\bar{\delta}_i^{(1)})}{\partial \bar{\delta}_i^{(1)}} = & \sum_{w_{ij}^{(1)}=1} \sum_k \sum_l \phi_{ik}^{(1)} \phi_{jl}^{(1)} \log b_{kl} \\ & - \sum_{w_{ij}^{(1)}=0} \sum_k \sum_l \phi_{ik}^{(1)} \phi_{jl}^{(1)} b_{kl}^{1+\bar{\delta}_i^{(1)}+\bar{\delta}_j^{(1)}} \log b_{kl} - \lambda + \frac{1}{\Sigma} (\bar{\delta}_i^{(2)} - \bar{\delta}_i^{(1)}) \end{aligned} \quad (5)$$

Similarly, the updated formula of other time steps can be obtained:  $t \in [2, T-1]$ :

$$\begin{aligned} \phi_{ik}^{(t)} \propto \exp\{ & \sum_{w_{ij}^{(t)}=1} (1 + \bar{\delta}_i^{(t)} + \bar{\delta}_j^{(t)}) \sum_l \phi_{jl}^{(t)} \log b_{kl} - \sum_{w_{ij}^{(t)}=0} \sum_l \phi_{jl}^{(t)} b_{kl}^{1+\bar{\delta}_i^{(t)}+\bar{\delta}_j^{(t)}} \\ & + \sum_l \phi_{il}^{(t-1)} \log A_{lk} + \sum_l \phi_{il}^{(t+1)} \log A_{kl} \} \end{aligned} \quad (6)$$

$$\begin{aligned} \frac{\partial \mathcal{O}(\bar{\delta}_i^{(t)})}{\partial \bar{\delta}_i^{(t)}} = & \sum_{w_{ij}^{(t)}=1} \sum_k \sum_l \phi_{ik}^{(t)} \phi_{jl}^{(t)} \log b_{kl} - \sum_{w_{ij}^{(t)}=0} \sum_k \sum_l \phi_{ik}^{(t)} \phi_{jl}^{(t)} b_{kl}^{1+\bar{\delta}_i^{(t)}+\bar{\delta}_j^{(t)}} \log b_{kl} \\ & - \frac{1}{\Sigma} (\bar{\delta}_i^{(t)} - \bar{\delta}_i^{(t-1)}) + \frac{1}{\Sigma} (\bar{\delta}_i^{(t+1)} - \bar{\delta}_i^{(t)}) \end{aligned} \quad (7)$$

$t = T$ :

$$\begin{aligned} \phi_{ik}^{(T)} \propto \exp\{ & \sum_{w_{ij}^{(T)}=1} (1 + \bar{\delta}_i^{(T)} + \bar{\delta}_j^{(T)}) \sum_l \phi_{jl}^{(T)} \log b_{kl} \\ & - \sum_{w_{ij}^{(T)}=0} \sum_l \phi_{jl}^{(T)} b_{kl}^{1+\bar{\delta}_i^{(T)}+\bar{\delta}_j^{(T)}} + \sum_l \phi_{il}^{(T-1)} \log A_{lk} \} \end{aligned} \quad (8)$$

$$\begin{aligned} \frac{\partial \mathcal{O}(\bar{\delta}_i^{(T)})}{\partial \bar{\delta}_i^{(T)}} &= \sum_{w_{ij}^{(T)}=1} \sum_k \sum_l \phi_{ik}^{(T)} \phi_{jl}^{(T)} \log b_{kl} \\ &- \sum_{w_{ij}^{(T)}=0} \sum_k \sum_l \phi_{ik}^{(T)} \phi_{jl}^{(T)} b_{kl}^{1+\bar{\delta}_i^{(T)}+\bar{\delta}_j^{(T)}} \log b_{kl} - \frac{1}{\Sigma} (\bar{\delta}_i^{(T)} - \bar{\delta}_i^{(T-1)}) \end{aligned} \tag{9}$$

**Algorithm 1.** Inference for DPSBM

**Input:** Initialization for model parameters  $\pi, B, A$  and variational parameters  $\phi^{(t)}, \bar{\delta}^{(t)}$  for  $t \in [1, T]$ ; the number of communities  $K$ ; stop criterion  $\varepsilon$ .

**Output:**  $\bar{\delta}^{(t)*}, \pi^*, B^*, A^*$  and  $z^*$

- 1: Compute variational likelihood  $\mathcal{L}^{new}$  by (3).
- 2: **repeat**
- 3:    $\mathcal{L}^{old} = \mathcal{L}^{new}$ .
- 4:   **for** every time  $t$  **do**
- 5:     **E-step**
- 6:     update  $\phi$  via (4)(6)(8).
- 7:     update  $\bar{\delta}$  by coordinate gradient ascend and gradient is given by (5)(7)(9).
- 8:     **M-step**
- 9:     update  $\pi$  via (10).
- 10:    update  $A$  via (11).
- 11:    update  $B$  by coordinate gradient ascend and gradient is given by (12).
- 12:    Compute variational likelihood  $\mathcal{L}^{new}$  with updated parameters by (3).
- 13:    **end for**
- 14: **until**  $|\mathcal{L}^{new} - \mathcal{L}^{old}| < \varepsilon$
- 15: **for** every time  $t$  **do**
- 16:    predict the cluster  $z_i^{(t)}$  of each node via (2).
- 17: **end for**

**4.2 M-Step**

We fixed the variational parameters  $(\phi, \delta)$  and maximized ELBO to obtain the model parameters  $(\pi, B, A)$ .

The optimal  $\pi$  is given by:

$$\pi_k \propto \sum_i \phi_{ik}^{(1)} \tag{10}$$

and the optimal  $A$  is given by:

$$A_{lk} \propto \sum_{t=2}^T \sum_i \phi_{il}^{(t-1)} \phi_{ik}^{(t)} \tag{11}$$

The optimal  $B$  is obtained by gradient ascend, and the gradient is computed by

$$\frac{\partial \mathcal{O}(b_{kl})}{\partial b_{kl}} = \frac{\sum_{t=1}^T \sum_{w_{ij}^{(t)}=1} (1 + \bar{\delta}_i^{(t)} + \bar{\delta}_j^{(t)}) \phi_{ik}^{(t)} \phi_{jl}^{(t)}}{b_{kl}} - \sum_{t=1}^T \sum_{w_{ij}^{(t)}=0} (1 + \bar{\delta}_i^{(t)} + \bar{\delta}_j^{(t)}) \phi_{ik}^{(t)} \phi_{jl}^{(t)} \bar{\delta}_i^{(t)} + \bar{\delta}_j^{(t)} \quad (12)$$

The pseudocode for algorithm is summarized as shown in Algorithm 1.

### 4.3 Complexity Analysis

In this section, we analysis roughly the complexity of Algorithm 1. The complexity of the inference algorithm depends on the following three steps. The complexity of the step which updates  $\phi$  is  $(TK^2n^2)$ , where  $T$  is the number of snapshots,  $n$  is the number of nodes in the network, and  $K$  is the number of communities. The complexity of the step which updates  $\delta$  is  $O(TK^2Cn^2)$ , where  $C$  is the number of iterations for gradient ascend. The complexity of the step which calculates the ELBO is  $O(TK^2n^2)$ . As a result, The complexity of the inference algorithm is  $O(TK^2n^2)$ . In theory, the complexity is  $O(n^2)$ . Since most real networks are sparse, we can improve efficiency and reduce running time and complexity by using negative sampling and parallelism in actual operation.

## 5 Experiments

In this section, we conduct several experiments to evaluate our proposed model. Firstly, we compare our model with four dynamic community detection methods in simulated datasets with homogeneous degree distribution, and prove that our method performed well in datasets with homogeneous degree distribution. Secondly, we test our model in simulated and real-world datasets with heterogeneous degree distribution, proving that our method is superior to the benchmark method in terms of community detection and evolution tracking, and verifying the interpretability and effectiveness of the popularity in our model. Thirdly, we analyze the hyperparameters sensitivity of the model. Finally, we apply the model to DBLP dataset and analyze the effect of community detection and evolution tracking. The experimental results verify the effectiveness of the model in the real dataset. At the same time, our method can reveal some meaningful insights that cannot be explained by other methods.

### 5.1 Performance Metrics

The performance metrics we use is the normalized mutual information (NMI), which is widely used in various fields [3, 18]. NMI is used when there exists ground truth, and measures the similarity between a given community partition and the ground truth. We use  $C = \{C_1, \dots, C_K\}$  to represent the true community

partition, and  $C' = \{C'_1, \dots, C'_K\}$  to represent the community partition to be evaluated, where  $C_k$  or  $C'_k$  is the collection of all nodes of the  $k$ th true community or the community to be evaluated. NMI is defined as:

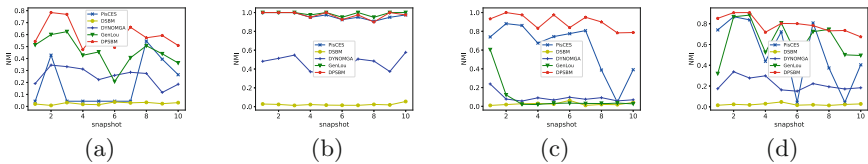
$$NMI(C, C') = \frac{\sum_{C, C'} p(C, C') \log \frac{p(C, C')}{p(C)p(C')}}{\max(H(C)H(C'))} \tag{13}$$

Where,  $H(C)$  and  $H(C')$  are the entropies of the community  $C$  and  $C'$ . The value of NMI is between 0 and 1. The higher the value of NMI is, the more similar the given community partition is to the true community partition, which indicates that the method is better.

### 5.2 Experiments on Simulated Datasets with Homogeneous Degree Distribution

The dataset we use is Facetnet, the benchmark adopted by Lin et al. [13], which was extended by the dataset proposed by Girvan and Newman. The dataset contains 10 snapshots and 128 nodes, which are divided into 4 communities with 32 nodes for each community. The average degree of nodes is *avgDegree*, and the degrees of nodes are almost uniformly distributed. Each node generates a number  $z$  of connections with nodes in other communities. With the increase of  $z$ , the noise level of the network also increases. In each time step from 2 to 10, several nodes in each community are randomly selected to leave the original community and to join the other three communities, and the number of nodes to leave is represented by  $nC$ .

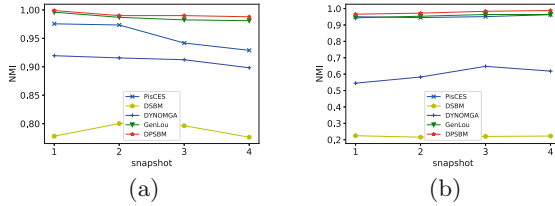
We test our method using simulated datasets generated with 4 different sets of parameters, and use NMI to compare our method with DSBM [19], DYNMOGA [2], GenLouvain [10] and PisCES [14]. The results show that our method is superior to the other four methods in the detection of community structure in the dynamic network with different noise level, different average degree and different transfer probability (see Fig. 2).



**Fig. 2.** The NMI on Facetnet simulated dataset: (a)  $z = 4, nC = 9, aD = 16$ ; (b)  $z = 4, nC = 9, aD = 20$ ; (c)  $z = 5, nC = 3, aD = 20$ ; (d)  $z = 5, nC = 9, aD = 20$ . The red line represents DPSBM, and the yellow, blue, green and turquoise lines represent DSBM, DYNMOGA, GenLouvain and PisCES respectively. (Color figure online)

### 5.3 Experiments on Simulated Datasets with Heterogeneous Degree Distribution

The simulated dataset with heterogeneous degree distribution we use is ASONAM, that is the benchmark dataset proposed by Green et al. in 2010 [4]. Several basic events in the dynamic process are introduced to make the simulated datasets more similar to the real-world datasets. We select two events to generate the networks. Each network contains 4 snapshots and 1000 nodes, with an average degree of 15 and a maximum degree of 50. The number of communities ranges from 20 to 50, and the probability of edges between communities is 0.2. The node degree of this dataset obeys the power-law distribution.

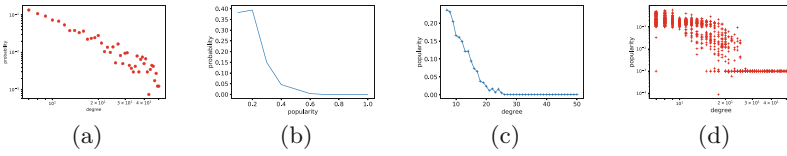


**Fig. 3.** The NMI on ASONAM dataset: (a) expansion and contraction; (b) swith. The red line represents DPSBM, and the yellow, blue, green and turquoise lines represent DSBM, DYNMOGA, GenLouvain and PisCES respectively. (Color figure online)

As can be seen from Fig. 3, our method has a good effect on the network with heterogeneous degree distribution, superior to the baseline method. At the same time, the line of our results is relatively smoother, which indicates that our detection results are more stable.

Figure 4(a) shows the degree distribution of the network, which indicates that the network has an important power-law distribution and scale-free property. Figure 4(b) shows the distribution of popularity, when popularity is relatively large, the probability is smaller, but we notice that when popularity is lower than about 0.2, the situation is somewhat different, which is the probability is relatively stable and even has a decline. Figure 4(c) and (d) show that there is a general negative correlation between degree and popularity. When the degree of nodes becomes large enough, the popularity value tends to a relatively small stable value. It is worth noting that in the current generation model with popularity [11, 14, 17], popularity always increases with the increase of degree. Thus, when there exists a node with a very large degree, even if there is a node with a very small degree in the network, the probability of connection between them will be abnormally high. This is not consistent with the actual situation. The popularity in our method has a small influence on the probability of node connection, when the degree of nodes is relatively large, while the effect on it is larger, when the degree is smaller, which indicates that our model can correct the deviation of connection.

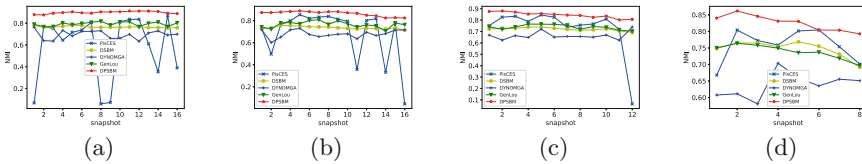




**Fig. 4.** Degree, popularity and correlation analysis of a simulated network. (a) degree distribution; (b) popularity distribution; (c) the relationship between the mean of the popularity and the degree; (d) correlation analysis of degree and popularity.

### 5.4 Experiments on Real-World Datasets

The real-world dataset we use is KIT-email data, which is an email network where nodes represent senders and recipients, and edges represent the connections between them. We assume that this network is undirected and unweighted. The data are divided into several time snapshots at intervals of 2, 3, 4, 6 months. The number of nodes in the network is between 138 and 231, and the number of communities is between 23 and 27.



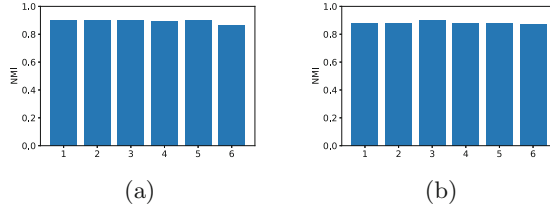
**Fig. 5.** The NMI on the email networks: (a) the interval is 2 months and  $T = 16$ ; (b) the interval is 3 months and  $T = 16$ ; (c) the interval is 4 months and  $T = 12$ ; (d) the interval is 6 months and  $T = 8$ . The red, yellow, blue, green and turquoise lines represent DPSBM, DSBM, DYNMOGA, GenLouvain and PisCES respectively. (Color figure online)

We use NMI to evaluate the performance of the different methods and compare the proposed method with the results of DSBM, DYNMOGA, GenLouvain and PisCES. As we can see, our method is superior to the other four methods in all four sets of data. We attribute this advantage to popularity and a more detailed characterization of scale-free properties in real networks. At the same time, our result curve is relatively smooth and there is no abrupt change in the adjacent snapshots, which indicates that our method can not only better discover the real community structure but also have a very stable effect (see Fig. 5).

### 5.5 Hyperparameters Sensitivity Analysis

We study the effect of hyperparameters on the model. Figure 6 shows the performance of DPSBM at different values of  $\lambda$  and  $\Sigma$ . The value of NMI is the mean of the NMI of all the snapshots. We can see that the values of  $\lambda$  and  $\Sigma$

have little impact on the results, which proves the robustness of DPSBM. We only show the results on the network with 2-month interval, and other networks have similar results, so are not shown here.



**Fig. 6.** The NMI value of our method under different values of  $\lambda$  and  $\Sigma$ . (a)  $\lambda = 0.01, 0.05, 0.1, 0.2, 0.5, 1$ ; (b)  $\Sigma = 0.01, 0.02, 0.05, 0.1, 0.2, 0.5$ .

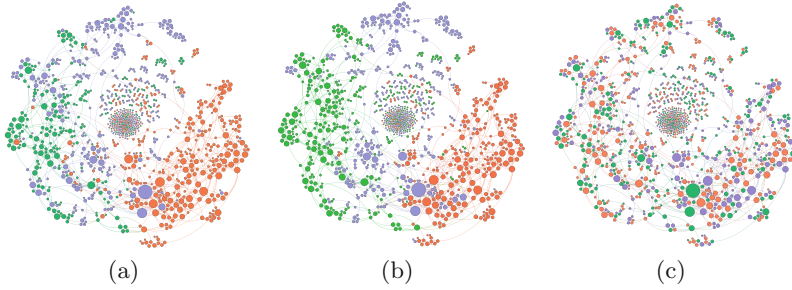
## 5.6 Case Study

We use the public DBLP dataset to verify the validity of DPSBM in practical applications. We select data from three major fields in the dataset—data Mining (DM), database (DB), and artificial intelligence (AI), which contains 1,163 nodes and 3 communities. We divided this dynamic network into 4 time snapshots.

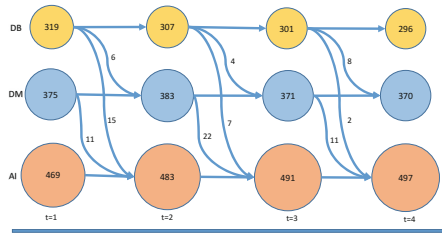
Figure 7 shows the clustering results of DPSBM and DSBM on the DBLP network. DSBM tends to put nodes with similar sizes in the same community, DPSBM corrects this deviation by introducing popularity, and thus brings the result closer to the ground truth. There are some differences between DPSBM and the ground truth, because the cooperative relationship in DBLP data not only depends on the author field but also has some irregular or personal factors.

We use Fig. 8 to show the community evolution in the DBLP network in four snapshots. It can be seen that most of the transfers occur from DB to DM and AI, and from DM to AI. As time goes by, the size of DB is decreasing while the size of AI is increasing, which is consistent with the development trend of the research field in reality. In addition to the study of community evolution, we also focus on some meaningful transfer individuals. We found that some of the authors with a higher degree also transferred, which also led to the partial transfer of other cooperative members. This result has certain practical significance, because, the transfer of influential authors, to some extent, indicates the research trend, and will have an impact on the research direction of other authors. We have verified the existence of these transfers by referring to the relevant data of the authors.

We accurately identified the community structures and the transfer of researchers without any labels or additional information, just through the collaboration between researchers, which shows that DPSBM can only use the network structure to find meaningful structures and changes. In the real world, it is often difficult to obtain attribute information other than structure information. For such imperfect data, DPSBM has important application value in the real world.



**Fig. 7.** Clustering results on DBLP network. (a) Ground truth. (b) DPSBM. (c) DSBM. Where, the node size represents the degree of the node, and the color represents the community, green for data mining, orange for database and blue for AI. (Color figure online)



**Fig. 8.** Community evolution for DBLP dataset

## 6 Conclusion

This paper proposes a unified probabilistic generation model to detect the community structure and track the community evolution in temporal complex networks. The framework integrates the heterogeneity of node degree to model the dynamic complex network and describes the scale-free characteristics in the real world, so as to correct the deviation of existing methods; an effective variational EM algorithm is proposed to optimize the objective function; the proposed algorithm can be well applied in practice. Extensive experiments show that our method outperforms the baseline algorithms of community detection in dynamic networks both in simulated and real-world datasets, and can explain some phenomena of real-world networks well. In the future, we will optimize the variational EM algorithm to make it more suitable for large-scale data.

**Acknowledgments.** This work was supported by the National Key R&D Program of China (2018YFC0809800, 2016QY15Z2502-02, 2018YFC0831000), the National Natural Science Foundation of China (91746107, 51438009, U1736103), and Tianjin Science and Technology Development Strategic Research Project (17ZLZDZF00430).

## References

1. Airoldi, E.M., Blei, D.M., Fienberg, S.E., Xing, E.P., Jaakkola, T.: Mixed membership stochastic block models for relational data with application to protein-protein interactions. In: Proceedings of the International Biometrics Society Annual Meeting, vol. 15 (2006)
2. Folino, F., Pizzuti, C.: An evolutionary multiobjective approach for community discovery in dynamic networks. *IEEE Trans. Knowl. Data Eng.* **26**(8), 1838–1852 (2014)
3. Gong, Y., Xu, W.: *Machine Learning for Multimedia Content Analysis*, vol. 30. Springer, Heidelberg (2007). <https://doi.org/10.1007/978-0-387-69942-4>
4. Greene, D., Doyle, D., Cunningham, P.: Tracking the evolution of communities in dynamic social networks. In: 2010 International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 176–183. IEEE (2010)
5. Holland, P.W., Laskey, K.B., Leinhardt, S.: Stochastic blockmodels: first steps. *Soc. Netw.* **5**(2), 109–137 (1983)
6. Holland, P.W., Leinhardt, S.: Local structure in social networks. *Sociol. Methodol.* **7**, 1–45 (1976)
7. Hopcroft, J., Khan, O., Kulis, B., Selman, B.: Tracking evolving communities in large linked networks. *Proc. Natl. Acad. Sci.* **101**(suppl 1), 5249–5253 (2004)
8. Jin, D., Chen, Z., He, D., Zhang, W.: Modeling with node degree preservation can accurately find communities. In: AAAI, pp. 160–167 (2015)
9. Jin, D., Wang, H., Dang, J., He, D., Zhang, W.: Detect overlapping communities via ranking node popularities. In: AAAI, pp. 172–178 (2016)
10. Jutla, I.S., Jeub, L.G., Mucha, P.J.: A generalized Louvain method for community detection implemented in matlab (2011). <http://netwiki.amath.unc.edu/GenLouvain>
11. Karrer, B., Newman, M.E.: Stochastic blockmodels and community structure in networks. *Phys. Rev. E* **83**(1), 016107 (2011)
12. Lee, P., Lakshmanan, L.V., Milios, E.E.: Incremental cluster evolution tracking from highly dynamic network data. In: 2014 IEEE 30th International Conference on Data Engineering (ICDE), pp. 3–14. IEEE (2014)
13. Lin, Y.R., Chi, Y., Zhu, S., Sundaram, H., Tseng, B.L.: FacetNet: a framework for analyzing communities and their evolutions in dynamic networks. In: Proceedings of the 17th international conference on World Wide Web, pp. 685–694. ACM (2008)
14. Liu, W., Saganowski, S., Kazienko, P., Cheong, S.A.: Using machine learning to predict the evolution of physics research. arXiv preprint [arXiv:1810.12116](https://arxiv.org/abs/1810.12116) (2018)
15. Tang, X., Yang, C.C.: Detecting social media hidden communities using dynamic stochastic blockmodel with temporal Dirichlet process. *ACM Trans. Intell. Syst. Technol. (TIST)* **5**(2), 36 (2014)
16. Wasserman, S., Faust, K.: *Social Network Analysis: Methods and Applications*, vol. 8. Cambridge University Press, Cambridge (1994)
17. Wilson, J.D., Stevens, N.T., Woodall, W.H.: Modeling and detecting change in temporal networks via a dynamic degree corrected stochastic block model. arXiv preprint [arXiv:1605.04049](https://arxiv.org/abs/1605.04049) (2016)
18. Xu, W., Gong, Y.: Document clustering by concept factorization. In: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 202–209. ACM (2004)

19. Yang, T., Chi, Y., Zhu, S., Gong, Y., Jin, R.: Detecting communities and their evolutions in dynamic social networks—a Bayesian approach. *Mach. Learn.* **82**(2), 157–189 (2011)
20. Zhang, G., Jin, D., Gao, J., Jiao, P., Fogelman-Soulié, F., Huang, X.: Finding communities with hierarchical semantics by distinguishing general and specialized topics. In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 3648–3654. AAAI Press (2018)



# In Good Company: Efficient Retrieval of the Top- $k$ Most Relevant Event-Partner Pairs

Dingming Wu<sup>1</sup>(✉), Yi Zhu<sup>1</sup>, and Christian S. Jensen<sup>2</sup>

<sup>1</sup> College of Computer Science and Software Engineering,  
Shenzhen University, Shenzhen, China  
dingming@szu.edu.cn, zhuyi2016@email.szu.edu.cn

<sup>2</sup> Department of Computer Science, Aalborg University, Aalborg, Denmark  
csj@cs.aau.dk

**Abstract.** The proliferation of event-based social networking (ESBN) motivates a range of studies on topics such as event, venue, and friend recommendation and event creation and organization. In this setting, the notion of event-partner recommendation has recently attracted attention. When recommending an event to a user, this functionality allows recommendation of partner with whom to attend the event. However, existing proposals are push-based: recommendations are pushed to users at the system's initiative. In contrast, EBSNs provide users with keyword-based search functionality. This way, users may retrieve information in pull mode. We propose a new way of accessing information in EBSNs that combines push and pull, thus allowing users to not only conduct ad-hoc searches for events, but also to receive partner recommendations for retrieved events. Specifically, we define and study the top- $k$  event-partner ( $k$ EP) pair retrieval query that integrates event-partner recommendation and keyword-based search for events. The query retrieves event-partner pairs, taking into account the relevance of events to user-supplied keywords and so-called together preferences that indicate the extent of a user's preference to attend an event with a given partner. In order to compute  $k$ EP queries efficiently, we propose a rank-join based framework with three optimizations. Results of empirical studies with implementations of the proposed techniques demonstrate that the proposed techniques are capable of excellent performance.

## 1 Introduction

The recent proliferation of the event-based social networking (EBSN), as exemplified by Meetup<sup>1</sup> and Eventbrite<sup>2</sup>, has not gone unnoticed in the research community, where substantial efforts have been devoted to the recommendation of events [5, 12, 23, 24, 33], venues [2, 14–16, 20], and friends [17, 28, 31]. Unlike previous recommendation techniques that each focus on recommending only one type

<sup>1</sup> <http://www.meetup.com>.

<sup>2</sup> <http://www.eventbrite.com>.

of items (either events or friends), event-partner recommendation [27] aims to recommend events together with partners to users. The rationale is that attending an event with a partner may be more attractive to a user than attending the event alone. Our put differently, a user may not attend an event if the user has to do so alone. However, no matter which recommendation technique is used, users receive information in a push mode, and the provided information is only related to the users' historical data. Yes, EBSNs also provide search services that allow users to retrieve event information in response to query keywords. This way users retrieve information in pull mode according to specified query keywords.

We propose a new way of accessing information in the EBSNs that combines push and pull modes, thus allowing users not only to conduct ad-hoc event search, but also to receive recommended partners for retrieved events. To achieve this goal, one may consider extending existing methods by first recommending events to a user and then filter out irrelevant events according to given keywords. However, this approach may produce empty results, since recommended events are usually based on a user's historical information while given keywords are ad hoc and may not align with the historical data. We adopt a different tack: we first retrieve relevant events w.r.t. given keywords, and then, for each relevant event, we find an appropriate partner.

Hence, we propose a new kind of EBSN query that takes advantage of both recommendation and search techniques. Taking into account user-specified keywords and a user's historical data, it retrieves event-partner pairs, such that the events are relevant to the keywords and such that the query user is willing to attend the events with the suggested partners. For instance, user "Mary" wants to attend a "rock concert". The query we propose retrieves  $k$  events relevant to "rock concert" and suggests a partner for each event. This of query is called the *top- $k$  event-partner ( $kEP$ ) pair retrieval query*. It differs from keyword queries that retrieve relevant events without partners, and is also differs from event-partner recommendation based on only the historical data, where ad-hoc query keywords are not taken into consideration.

The  $kEP$  query can be modeled as a top- $k$  join query that returns the  $k$  join results (pairs of events and users) with the highest scores. A straightforward method to process the  $kEP$  query is to first join events and the users and then, for each event  $e$ , choose the pair  $(e, u_i)$  with the highest so-called together preference as a result candidate. Next, all the candidates are ranked according to a scoring function, and the  $k$  candidates that score the highest are returned as the results. However, this method is inefficient, since all possible combinations of events and users are considered in the join process. One may consider to extend the rank-join algorithm [10] to answer the  $kEP$  queries, yielding a method that is more efficient than the straightforward method just described. The idea is to scan input events and users ordered according to their scoring predicates. While a scoring function might use textual relevance of descriptions of events to the query keywords, there is no obvious scoring predicate for the partners with which to attend events.

We propose a rank-join based framework for computing  $k$ EP queries where the scoring predicate for the partners (users) is the number of events that they attended. Intuitively, users who have attended many events tend to have high so-called together preferences, to be detailed in the next section. Two representative join strategies, nested loop join and ripple join [7], are studied within the framework. An empirical study offers evidence that the ripple join is better than the nested loop join. To further improve the performance of the framework, three optimizations are proposed: (1) an unpromising-event pruning technique that removes encountered events that cannot enter the result, (2) a key partner technique that quickly identifies results by exploiting a property of the scoring function, and (3) an efficient partner computation technique that reduces the computational cost of finding the partner with the highest together preference for an event. To evaluate the proposed framework and optimizations, we conduct experiments with prototype implementations of the proposed techniques using real data. The results offer insight into the properties of the techniques and indicate that the paper’s proposal is useful in practice.

The rest of this paper is organized as follows. Section 2 formally defines the top- $k$  event-partner retrieval problem. Section 3 presents the framework. The three optimizations are detailed in Sect. 4. We report on a performance evaluation in Sect. 5. Finally, we cover related work in Sect. 6 and offer conclusions and research directions in Sect. 7.

## 2 Problem Definition

An event-based social network (EBSN) can be modeled as a bipartite graph  $G = (U, E, R)$ , where  $U$  represents a set of users,  $E$  models a set of events posted by the users, and  $R \subseteq U \times E$  is set of participation-relationships between users and events, i.e.,  $r = (u, e) \in R, u \in U, e \in E$ . Each event  $e \in E$  is associated with a text document  $e.\psi$  that describes the content and features of the event. Specifically, we will assume that a document is represented by a term vector [25]. The members of an event  $e$  are the users  $u$  who have joined  $e$ :  $\{u \mid (u, e) \in R\}$ .

The **together preference** [27]  $p(u^*, e, u)$  measures the probability that user  $u^*$  is willing to attend event  $e$  with user  $u$ ,  $u^* \neq u$ , i.e., user  $u$  is willing to be a partner of  $u^*$  at event  $e$ . Its definition is given in Eq. 1, and it is motivated by two observations. First, a user may wish to attend an event that is similar to events that the user has previously participated in. Second, people tend to join an event with a partner with whom they share common interests. In our scenario, the common interests are captured by the common events that two people have participated in.

$$p(u^*, e, u) = \frac{\sum_{e_i \in N(u^*, e)} s(e, e_i) \cdot b(u^*, e_i, u)}{\sum_{e_i \in N(u^*, e)} s(e, e_i)} \quad (1)$$

Function  $p(u^*, e, u)$  takes three arguments, i.e., a target user  $u^*$ , an event  $e$ , and a partner user  $u$ . The range of  $p(u^*, e, u)$  is  $[0, 1]$ . Large values of  $p(u^*, e, u)$



indicate that target user  $u^*$  is very likely to attend event  $e$  with partner user  $u$ . In the definition,  $N(u^*, e)$  is the neighborhood of a pair of a user and an event  $(u^*, e)$ , which is the set of events  $e_i$  that satisfy the following two conditions: (1) user  $u^*$  has attended event  $e_i$ , and (2) the similarity  $s(e, e_i)$  between the documents of events  $e$  and  $e_i$  is no less than a threshold  $\tau$ . We define the similarity  $s(e, e_i)$  as the cosine similarity. However, the proposed method is independent of the choice of the similarity measure. Any reasonable similarity function can be adopted easily. Given a partner user  $u$ , a target user  $u^*$ , and an event  $e_i$ ,  $b(u^*, e_i, u) = 1$  if  $u^*$  and  $u$  have participated in event  $e_i$ , i.e.,  $(u^*, e_i) \in R$  and  $(u, e_i) \in R$ ; otherwise,  $b(u^*, e_i, u) = 0$ . The denominator is the sum of the similarities between event  $e$  and each event  $e_i$  in the neighborhood  $N(u^*, e)$ . The numerator sums up the similarities between event  $e$  and the event  $e_i$  in neighborhood  $N(u^*, e)$  that  $u$  has participated in. The together preference is not symmetric, i.e.,  $p(u^*, e, u) \neq p(u, e, u^*)$ . If  $N(u^*, e) = \emptyset$ , the together preference  $p(u^*, e, u)$  is undefined, which means that no partner can be recommended for  $u^*$  for participating in event  $e$ . The together preference can be interpreted in two phases. First, an event  $e$  is taken as a candidate event for user  $u^*$  if some of  $u^*$ 's previously attended events are similar to  $e$ , i.e,  $N(u^*, e) \neq \emptyset$ . Second, a user  $u$  is considered a candidate partner w.r.t.  $u^*$  and  $e$  if  $u$  has participated in events in neighborhood  $N(u^*, e)$ .

*Example 1.* Consider the EBSN in Fig. 1, where there are five users  $U = \{u_1, u_2, \dots, u_5\}$  and five events  $E = \{e_1, e_2, \dots, e_5\}$ . The participation relationship  $R$  between users and events is given by the edges. Table 1 shows the term vectors of the documents of the events, and Table 2 shows the similarities between the documents of the events. Given  $\tau = 0.3$ , the neighborhood of user-event pair  $(u_4, e_3)$ ,  $N(u_4, e_3)$  is  $\{e_2, e_4, e_5\}$  because (i) user  $u_4$  has attended events  $e_1, e_2, e_3, e_4$ , and  $e_5$ , and (ii) the similar (having similarity no less than 0.3) events of  $e_3$  are  $e_2, e_4$ , and  $e_5$ . The together preference  $p(u_4, e_3, u_3) = (0.5 + 0.6)/(0.5 + 0.3 + 0.6) = 0.79$ , since  $u_3$  has attended events  $e_2$  and  $e_5$  in neighborhood  $N(u_4, e_3)$ .

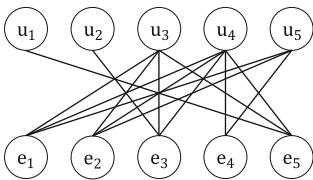


Fig. 1. Example EBSN

Table 1. Term vectors

Event	Term vector					
$e_1$	$t_2$	$t_3$	$t_4$	$t_9$		
$e_2$	$t_1$	$t_3$	$t_7$	$t_8$		
$e_3$	$t_1$	$t_3$	$t_5$			
$e_4$	$t_2$	$t_3$	$t_6$	$t_9$		
$e_5$	$t_1$	$t_3$	$t_4$	$t_5$	$t_6$	

Table 2. Similarities

	$e_1$	$e_2$	$e_3$	$e_4$	$e_5$
$e_1$	1	0.2	0.2	0.7	0.3
$e_2$	0.2	1	0.5	0.2	0.3
$e_3$	0.2	0.5	1	0.3	0.6
$e_4$	0.7	0.2	0.3	1	0.3
$e_5$	0.3	0.3	0.6	0.3	1

A **top- $k$  most relevant Event-Partner pair retrieval ( $k$ EP)** query  $Q = (k, u_q, \psi_q)$  takes three arguments: (i) a number  $k$  of requested event-partner pairs, (ii) a query user  $u_q$ , and (iii) a set of query keywords  $\psi_q$ . Let  $t(\psi_q, e, \psi)$  be the textual relevance (e.g., defined using language models [22]) of event  $e$  w.r.t. the query keywords  $\psi_q$ . For each  $e \in E$ , let  $u^e$  be the user who maximizes

$p(u_q, e, u)$ , i.e.,  $u^e = \arg \max_{u \in U} p(u_q, e, u)$ . The result of a  $k$ EP query contains  $k$  event-partner pairs  $(e, u)$  with the highest score  $f(u_q, \psi_q, e, u^e)$  (Eq. 2). The events in the result are distinct, but the same partner user may be paired with multiple events. The scoring function considers both textual relevance and the together preference. Ties are broken arbitrarily.

The scoring function used in the paper takes the form of a weighted sum of the textual relevance and the together preference. The techniques we propose are, however, applicable to any scoring function that is monotone in terms of both the textual relevance and the together preference. The  $k$ EP query tries to find event-partner pairs  $(e, u)$  such that the events are relevant to the query keywords and the query user is likely to participate in the events with partners.

$$f(u_q, \psi_q, e, u^e) = \alpha \cdot t(\psi_q, e, \psi) + (1 - \alpha) \cdot p(u_q, e, u^e)$$

$$s.t. \quad t(\psi_q, e) \in [0, 1] \quad \wedge \quad p(u_q, e, u^e) \in [0, 1] \quad (2)$$

In the EBSN, some events may occur periodically, e.g., weekly or monthly. A user may attend same such event multiple times. Hence, in the result of a  $k$ EP query, events may exist that have been attended previously by the query user. We do not exclude those events, since the query user is probably interested in them and may attend them again.

*Example 2.* Continuing Example 1, given a  $k$ EP query  $Q$  with  $k = 2$ ,  $u_q = u_4$ , and  $\psi_q = \{t_1, t_3\}$ , the textual relevance of each event w.r.t. the query keywords is shown in Table 3, and the neighborhoods of user-event pairs are shown in Table 4 (given  $\tau = 0.3$ ). Given the query user  $u_4$ , for each event  $e$ , we choose user  $u^e$  as the partner, given that the together preference  $p(u_4, e, u^e)$  exceeds the together preference of choosing any other user. Table 4 shows the partner user for each event and the corresponding together preference. Given  $\alpha = 0.5$ , the top-2 event-user pairs of query  $Q$  are  $(e_2, u_3)$  and  $(e_3, u_3)$  with score 0.85 and 0.8, respectively.

**Table 3.** Textual relevance

Event	$t(\psi_q, e, \psi)$
$e_1$	0.4
$e_2$	0.7
$e_3$	0.8
$e_4$	0.3
$e_5$	0.6

**Table 4.** Neighborhood

Event	$N(u_4, e)$	$u^e$	$p(u_4, e, u^e)$
$e_1$	$e_4, e_5$	$u_5$	0.7
$e_2$	$e_3, e_5$	$u_3$	1
$e_3$	$e_2, e_4, e_5$	$u_3$	0.79
$e_4$	$e_1, e_3, e_5$	$u_3$	1
$e_5$	$e_1, e_2, e_3, e_4$	$u_3$	0.8

### 3 Rank-Join Based Framework

We proceed to present the rank-join based framework for the processing of  $k$ EP queries. Section 3.1 presents the main data structures used in the framework. Section 3.2 explains the query processing algorithm, and Sect. 3.3 covers two join strategies in the framework.

### 3.1 Data Structures

The rank-join based framework includes two main data structures. One is a representation of the event-user graph  $G$  that is stored in the main memory. The other one is a disk-resident inverted index  $II_d$  that indexes the documents of all events in the EBSN. The inverted index consists of two main components: (1) a vocabulary of all distinct terms in the collection of documents and (2) a posting list for each term  $t$  in the vocabulary. Each posting list is a sequence of pairs  $(id, w)$ , where  $id$  identifies an event  $e$  whose document  $e.\psi$  contains term  $t$  and  $w$  is the weight of term  $t$  in document  $e.\psi$ .

### 3.2 Algorithm

Following the idea of the rank-join algorithm [10], the framework conducts a join operation on the ranked input events and users. The ranked input events are obtained by issuing a keyword query using the inverted index  $II_d$ . The relevant events are retrieved in descending order of their textual relevance. In contrast, there is no straightforward way to obtain the ranked input users. Following the definition of the together preference (Eq. 1), the framework uses a heuristic scoring predicate of the users, namely the number of events the users have attended. The motivation is that the users who have attended many events can be expected to have high together preferences w.r.t. query user  $u_q$  and event  $e$ . However, this heuristic falls short when a user has attended many events outside neighborhood  $N(u_q, e)$ . To avoid this situation, the framework retrieves the users based on two constraints, i.e., (1) the retrieved users should have attended at least one event in neighborhood  $N(u_q, e)$ , and (2) the users are retrieved in descending order of the number of events they have attended. Specifically, the set of users  $U(u_q, e)$  that is fed into the join process is constructed as follows. For each retrieved event  $e$ , its neighborhood  $N(u_q, e)$  is computed. Then, for each event in the neighborhood, its participants are obtained from the event-user graph. The user set  $U(u_q, e)$  is the union of the members of all the events in neighborhood  $N(u_q, e)$ . Next, the users in  $U(u_q, e)$  are sorted descendingly on the number of events they have attended.

The query processing algorithm in the framework borrows the idea of the TA (Threshold Algorithm) [3] and consumes the input events and users to generate candidate event-user pairs. A threshold  $T$  is maintained that is calculated by setting the textual relevance to that of the upcoming event and the together preference to 1 in the scoring function (Eq. 2). When the score of the  $k^{th}$  largest candidate pair is no less than  $T$ , the algorithm reports the obtained top- $k$  pairs. It is not difficult to prove that threshold  $T$  serves as an upper bound on the scores of the event-partner pairs that have not yet been considered, since the events are retrieved in descending order of the textual relevance and because the maximum value of the together preference is set to 1. Different join strategies can be adopted for the join in the algorithm.

### 3.3 Join Strategies

We consider two state-of-the-art join strategies in the framework, namely nested loop join and ripple join [7]. The nested loop join consists of two nested loops, where the outer loop consumes events in descending order of the textual relevance and the inner loop, executed for each (outer) event, consumes the users in  $U(u_q, e)$ . When the inner loop for an event finishes, the partner for the event with the highest together preference has been identified and this event-partner pair is output as a candidate.

In the ripple join, e.g., the “square” version, one previously unseen tuple (user or event) is retrieved from each of the two input lists in each step; these new tuples are joined with the previously seen tuples and with each other. As in the nested loop join, the events are retrieved in the descending order of the textual relevance. Unlike in the nested loop join, the users to be retrieved are organized in a priority queue sorted descendingly on the number of events they have attended. For each upcoming event  $e$ , the priority queue is updated dynamically by adding its user set  $U(u_q, e)$ . The square version consumes one event and one user at a time. In the empirical study, we also evaluate the performance of the rectangular version that consumes different numbers of events and users at a time.

## 4 Optimizations

Although the rank-join based framework take advantage of both the rank-join and the TA algorithm, it is still inefficient in some cases. For instances, it may take a long time to find the partner for an event if the number of users considered in the join process is large. In addition, before returning the top- $k$  pairs, unnecessarily many candidate pairs may have been produced, which incurs high computational cost. We develop three optimizations with the goal of improving the performance of the framework.

### 4.1 Unpromising-Event Pruning

This optimization reduces the computational cost by pruning events that will definitely not contribute to top- $k$  pairs. We derive a worst allowed together preference  $p_w(e)$  (Definition 1) for a newly retrieved event, which is a necessary condition of the event being able to contribute to the top- $k$  result. This value is a lower bound on the together preference of the events in the final result. We also derive a best possible together preference  $p_{ub}(e)$  (Definition 2) for a newly retrieved event, which estimates the highest possible together preference of the event with its partner.

**Definition 1. Worst Allowed Together Preference  $p_w(e)$ :** Given a query user  $u_q$  and query keywords  $\psi_q$ , let  $f_k$  be the score of the current  $k^{\text{th}}$  candidate event-partner pair. The worst allowed together preference  $p_w(e)$  of  $e$  is defined as  $p_w(e) = (f_k - \alpha \cdot t(\psi_q, e)) / (1 - \alpha)$ .

**Lemma 1.** *If  $\forall u \in (U \setminus \{u_q\})(p(u_q, e, u) \leq p_w(e))$ , event  $e$  cannot belong to the result.*

**Definition 2. Best Possible Together Preference  $p_{ub}(e)$ :** *Let  $N(u_q, e)$  be the neighborhood of  $u_q$  and  $e$ , and define  $m = \max_{u \in (U \setminus \{u_q\})\{|E_c| \mid E_c = N(u_q, e) \cap E_u\}}$ , where  $E_u$  is the set of events that user  $u$  have attended. The best possible together preference  $p_{ub}(e)$  of  $e$  is given as follows.*

$$p_{ub}(e) = \frac{\sum_{e_i \in TopM(u_q, e)} s(e, e_i)}{\sum_{e_i \in N(u_q, e)} s(e, e_i)}, \quad (3)$$

where  $|TopM(u_q, e)| = m$ ,  $TopM(u_q, e) \subseteq N(u_q, e)$ , and  $\forall e_i \in TopM(u_q, e) \forall e_j \in (N(u_q, e) \setminus TopM(u_q, e)) (s(e, e_i) \geq s(e, e_j))$ .

**Lemma 2.** *The best possible together preference  $p_{ub}(e)$  of  $e$  is an upper bound on the together preference of  $e$  with its partner.*

*Proof.* According to Eq. 1, the numerator of the together preference sums up the similarities between the events  $e_i$  that user  $u$  has attended in neighborhood  $N(u_q, e)$ . This can be rewritten as follows:  $\sum_{e_i \in N(u_q, e) \cap E_u} s(e, e_i)$ . Set  $TopM(u_q, e)$  contains the top- $m$  similar events in neighborhood  $N(u_q, e)$ . Since  $m = \max_{u \in (U \setminus \{u_q\})\{|E_c| \mid E_c = N(u_q, e) \cap E_u\}}$  is the maximum number of events attended by a user in  $N(u_q, e)$ , we have  $\forall u \in (U \setminus \{u_q\})(|TopM(u_q, e)| \geq |N(u_q, e) \cap E_u|)$ . It is easy to derive that  $\forall u \in (U \setminus \{u_q\})(\sum_{e_i \in N(u_q, e) \cap E_u} s(e, e_i) \leq \sum_{e_i \in TopM(u_q, e)} s(e, e_i))$ . The denominators of  $p_{ub}(e)$  and the together preference are the same. Hence, we have proven that  $\forall u \in (U \setminus \{u_q\})(p_{ub}(e) \geq p(u_q, e, u))$ .

*Example 3.* Given query user  $u_4$ , we illustrate how to compute the best possible together preference of event  $e_3$ . Neighborhood  $N(u_4, e_3) = \{e_2, e_4, e_5\}$  and  $m = 2$ . Then we have  $TopM(u_4, e_3) = \{e_2, e_5\}$ . The best possible together preference is calculated as  $p_{ub}(e_3) = (0.5 + 0.6)/(0.5 + 0.3 + 0.6) = 0.79$ .

Lemmas 1 and 2 present the properties of  $p_w(e)$  and  $p_{ub}(e)$ , respectively. Pruning Rule 1 below is based on Lemmas 1 and 2 and is able to prune unpromising events (that cannot contribute to pairs in the top- $k$  result). Thus it enables reducing the computational cost.

**Pruning Rule 1.** *Given query user  $u_q$  and query keywords  $\psi_q$ , for event  $e$ , if  $p_w(e) > p_{ub}(e)$ , event  $e$  cannot contribute to a result pair and can be pruned.*

## 4.2 Key Partner

It follows from the definition of the together preference (Eq. 1) that if a user  $u$  exists who has attended all events in neighborhood  $N(u_q, e)$ , the together preference  $p(u_q, e, u)$  equals 1, the maximum value. Thus, user  $u$  is the partner for event  $e$ . Based on this observation, we introduce the key partner set (Definition 3) of a user  $u$ . If a query user  $u_q$  has a key partner, any candidate event for  $u_q$  will

be paired with the key partner, and the together preference is 1 (Lemma 3). In other words, computing the top- $k$  event-partner pairs for a query user who has a key partner is efficient: the  $k$  events with the highest textual relevance are retrieved and paired with the key partner.

**Definition 3. Key Partner Set  $KP(u)$ :** Let  $E_u$  and  $E_{u'}$  be the sets of events attended by users  $u$  and  $u'$ , respectively. If  $E_u \subseteq E_{u'}$ , user  $u'$  is a key partner of user  $u$ . The key partner set is defined as  $KP(u) = \{u_i | \forall u_i \in U \setminus \{u\} (E_u \subseteq E_{u_i})\}$ .

**Lemma 3.** Given a query user  $u_q$ ,  $\forall e \in E \forall u \in KP(u_q) \forall u' \in U \setminus (KP(u_q) \cup \{u_q\}) (p(u_q, e, u) = 1 \geq p(u_q, e, u'))$ .

*Proof.* Since  $\forall u \in KP(u_q) (E_u \supseteq E_{u_q})$  and  $\forall e \in E (E_{u_q} \supseteq N(u_q, e))$ , we have  $\forall u \in KP(u_q) \forall e \in E (E_u \supseteq N(u_q, e))$  and derive that  $p(u_q, e, u) = 1$ . Straightforwardly,  $\forall u' \in U \setminus (KP(u_q) \cup \{u_q\}) (p(u_q, e, u) = 1 \geq p(u_q, e, u'))$ .

The key partner set of a query user may contain multiple users. According to the definition of the  $kEP$  query, the events in the result are distinct. Thus, we arbitrarily select one user from the key partner set for each event.

### 4.3 Efficient Partner Computation

In the framework, the operation of finding a partner for an event is expensive. In particular, the cost is high when user set  $U(u_q, e)$  considered in the join is large. The following optimization provides a way of finding the partner for an event without examining each user in set  $U(u_q, e)$ . The optimization uses an event-member list for each event that consists of pairs  $(u, num)$  sorted descendingly on  $num$ , which is the number of events attended by user  $u$ .

Table 5 shows the event-member lists of the five events in Fig. 1. For instance, event  $e_4$  has members  $u_4$  and  $u_5$ . User  $u_4$  has attended five events, and user  $u_5$  has attended three events.

**Table 5.** Event-member lists

Event	Member list
$e_1$	$(u_4, 5), (u_3, 4), (u_5, 3)$
$e_2$	$(u_4, 5), (u_3, 4), (u_5, 3)$
$e_3$	$(u_4, 5), (u_3, 4), (u_2, 1)$
$e_4$	$(u_4, 5), (u_5, 3)$
$e_5$	$(u_4, 5), (u_3, 4), (u_1, 1)$

Algorithm 1 shows the pseudo code of the efficient partner computation. It takes a query user  $u_q$ , an event  $e$ , and a neighborhood  $N(u_q, e)$  as arguments, and it returns the partner user  $u$  who maximizes the together preference  $p(u_q, e, u)$ .

Given neighborhood  $N(u_q, e)$ , the member lists of the events in the neighborhood are fetched (line 4). Function **GetNextPair**() chooses the pair  $(u, num)$ ,  $u \neq u_q$  with the largest  $num$  from the first elements of all fetched member lists. If multiple pairs have the same largest  $num$ , the pair from the member list of event  $e_i$  with the largest similarity  $s(e, e_i)$  is selected (line 6). The together preference  $p(u_q, e, u)$  is computed, and pair  $(u, num)$  is removed from each member list that contains it. If  $p(u_q, e, u)$  is larger than the together preference  $p_1$  of the current candidate partner  $u_p$ , user  $u$  is taken as the candidate partner (lines 7–10). Then function **GetNextPair**() is called again to obtain the next pair  $(u, num)$  that is used to compute an upper bound  $p_r$  (Lemma 4) on the together preference of the rest of the users in the fetched member lists (lines 12–14). If the together preference  $p_1$  of the current candidate partner  $u_p$  is no less than  $p_r$ , user  $u_p$  is the partner who maximizes  $p(u_q, e, u)$  and is returned (Pruning Rule 2). Otherwise, the algorithm repeats the above process.

---

**Algorithm 1.** ComputePartner( $N(u_q, e), u_q, e$ )
 

---

```

1:  $p_1 \leftarrow -1$ 
2:  $p_r \leftarrow +\infty$ 
3:  $u_p \leftarrow null$ 
4: Fetch the member list  $ml(e_i)$  of each event in  $N(u_q, e)$ 
5: while  $p_1 < p_r$  do ▷ Pruning Rule 2
6:    $(u, num) \leftarrow \mathbf{GetNextPair}()$ 
7:   if  $p_1 < p(u_q, e, u)$  then
8:      $p_1 \leftarrow p(u_q, e, u)$ 
9:      $u_p \leftarrow u$ 
10:  end if
11:  Remove  $(u, num)$  from the member list
12:   $(u, num) \leftarrow \mathbf{GetNextPair}()$ 
13:   $x \leftarrow \min\{num, |N(u_q, e)|\}$ 
14:   $p_r \leftarrow (\sum_{e_i \in TopX(u_q, e)} s(e, e_i)) / (\sum_{e_i \in N(u_q, e)} s(e, e_i))$ 
15: end while
16: return  $u_p$ 

```

---

**Lemma 4.** Given a query user  $u_q$  and an event  $e$ , let  $(u, num)$  be the pair returned by function **GetNextPair**() and define  $x = \min\{num, |N(u_q, e)|\}$ . Then set  $TopX(u_q, e)$  contains the top- $x$  events  $\{e_i\}$  in neighborhood  $N(u_q, e)$  with the largest similarity  $s(e, e_i)$ . An upper bound on the together preference of the users in the member lists of the events in  $N(u_q, e)$  is

$$p_r = \frac{\sum_{e_i \in TopX(u_q, e)} s(e, e_i)}{\sum_{e_i \in N(u_q, e)} s(e, e_i)} \quad (4)$$

*Proof.* Recall that (i) the pairs  $(u, num)$  in the member list are sorted descending on  $num$  and that (ii) function **GetNextPair**() returns the pair with the

largest  $num$  from the first elements in all member lists of the events in  $N(u_q, e)$ . This means that no user in the member lists of the events in  $N(u_q, e)$  can have attended more events than the returned  $num$ . Since  $x = \min\{num, |N(u_q, e)|\}$ , no user in the member lists of the events in  $N(u_q, e)$  has attended more than  $x$  events in  $N(u_q, e)$ . Given that set  $TopX(u_q, e)$  contains the top- $x$  events  $\{e_i\}$  in neighborhood  $N(u_q, e)$  with the largest similarity  $s(e, e_i)$ , then, for any user  $u_i$  in the member lists of the events in  $N(u_q, e)$ , we have  $p_r \geq p(u_q, e, u_i)$ .

**Pruning Rule 2** *In the context of Algorithm 1, let  $p_1$  be the together preference of the current candidate partner  $u_p$ . If  $p_1 \geq p_r$ , user  $u_p$  is returned as the partner of event  $e$ , and no other user in the fetched member lists can be the partner and are pruned.*

*Example 4.* Given query user  $u_4$  and event  $e_2$ , according to Table 4, neighborhood  $N(u_4, e_2) = \{e_3, e_5\}$ . According to Table 5, the pair returned by function **GetNextPair()** is  $(u_3, 4)$ . The together preference  $p(u_4, e_2, u_3) = 1$ . Pair  $(u_3, 4)$  is removed from the member list of each event in  $N(u_4, e_2)$ . User  $u_3$  is taken as the candidate partner. Next, **GetNextPair()** returns  $(u_2, 1)$ . We have  $x = 1$  and  $p_r = 0.5/(0.5 + 0.3) = 0.63$ . Since  $p(u_4, e_2, u_3) \geq p_r$ , no other user can have higher together preference than does user  $u_3$ . Finally, user  $u_3$  is returned as the partner for  $u_4$  at  $e_2$ .

## 5 Empirical Study

We proceed to cover a study of the performance of the proposed framework and its three optimizations. In the experiments, F-NLJ and F-NLJ\* denote the framework adopting the nested loop join without and with optimizations, respectively. F-RJ\* denotes the framework adopting the ripple join with optimizations.

### 5.1 Data and Queries

We have crawled a data set from Meetup<sup>3</sup> that contains 224,238 events and 7,822,965 users. The average number of members per event is 116. We have also downloaded the text descriptions (documents) of the events. The number of unique terms in the document collection is 519,885, and the average number of tokens per document is 72.

We generated 5 query sets, in which the number of keywords is 1, 2, 3, 4, and 5, respectively, taken from the data set. Each query set comprises 100 queries. Specifically, to generate a query, we randomly pick a user in the data set as the query user, and we randomly choose words from the document of a randomly selected event as the query keywords. We ensure that no query has an empty result.

<sup>3</sup> <http://www.meetup.com>.



## 5.2 Setup

All algorithms were implemented in Java, and a machine with an Intel(R) Xeon(R) CPU E5-2630 v2@2.60GHz and 128 GB main memory was used for the experiments. The document inverted index is implemented by Lucene<sup>4</sup> and is disk resident. The key partner sets of all users are kept in main memory. In the data set, 68% of all users have key partner sets. The user-event graph  $G$  is represented by adjacency lists and is stored in main memory. Since the structure of the member lists of the events is similar to the adjacency lists of the user-event graph, we extend the adjacency lists of the event nodes in  $G$  to include the number of events attended by each user, so that the member list of any event can be obtained from the user-event graph.

We study the effects of different parameters and set parameter default values as follows: the number  $k$  of requested event-partner pairs is 10; the number of query keywords is 3; parameter  $\alpha$  in the scoring function (Eq. 2) is 0.5. Some queries take long time to compute using the framework without optimizations. We set 20 s as a time limit. If the processing of any query exceeds 20 s, we stop the processing.

## 5.3 Performance Evaluation

### Tuning the Number of Retrieved Events and Users in the Ripple Join.

When using the ripple join, the numbers of retrieved events and users at a time affects the performance. We thus evaluate the framework using the ripple join when varying the numbers of retrieved events and users. It is observed that varying the number of retrieved events does not affect the performance, while varying the number of retrieved users does, as shown in Fig. 2. The runtime improves as the number of retrieved users is increased from 10 to 100, and it gets slightly worse when the number of retrieved users is increased from 100 to 150. The number of events and users involved in the computation and the number of pruned events exhibit similar behavior. Thus, in the following experiment, the number of retrieved users in the ripple join is set to 100.

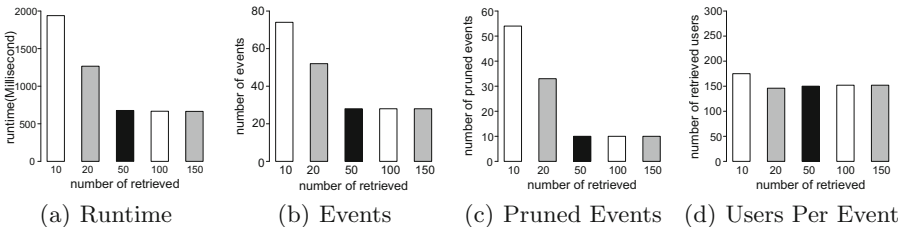


Fig. 2. Varying the number of retrieved users in ripple join

<sup>4</sup> <https://lucene.apache.org>.

**Varying the Number  $k$  of Requested Event-Partner Pairs.** Figure 3 shows the average runtime, the average number of retrieved events per query, the average number of pruned events, and the average number of retrieved users per event when varying  $k$ . The average runtime of the three approaches increases as  $k$  increases, since more and more relevant events are retrieved and more and more users are involved as  $k$  increases. The unpromising event pruning optimization prunes many events, as shown in Fig. 3(c). The efficient partner computation optimization reduces the number of retrieved users per event, as shown in Fig. 3(d). F-NLJ has almost the same number of retrieved events as each of F-NLJ\* and F-RJ\* (Fig. 3(b)). This is because F-NLJ has significantly more retrieved users per event than do F-NLJ\* and F-RJ\*, so that the processing time of some of the queries exceeds the 20s time limit, forcing those queries to stop. Thus, the framework with optimizations outperforms F-NLJ significantly in terms of runtime. F-RJ\* has slightly fewer pruned events than does F-NLJ\*, but it also has significantly fewer retrieved users per event than does F-NLJ\*, which means that the ripple join is better than the nested loop join. Hence, F-RJ\* outperforms F-NLJ\* in terms of runtime.

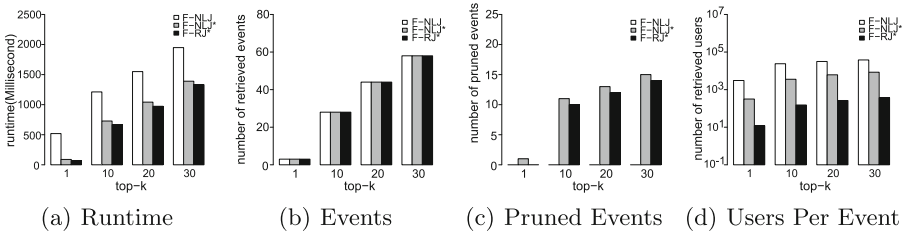


Fig. 3. Varying the number of requested event-partner pairs  $k$

**Varying the Number of Query Keywords.** Figure 4 shows the average runtime, the average number of retrieved events per query, the average number of pruned events, and the average number of retrieved users per event when varying the number of query keywords. It can also be seen that the unpromising event pruning and efficient partner computation optimizations are effective, cf. Figure 4(c) and (d). The number of events retrieved by F-NLJ is slightly lower than those of both F-NLJ\* and F-RJ\* (Fig. 4(b)). This occurs because the processing of some of the queries using F-NLJ take too long and are forced to stop. Overall, F-NLJ\* and F-RJ\* outperform F-NLJ significantly in terms of runtime.

**Varying  $\alpha$ .** Figure 5 reports on the finding when varying  $\alpha$  that specifies the weight of the textual relevance in the scoring function. The performance of the three approaches get better (shorter runtime, fewer retrieved events and users) as  $\alpha$  increases. The reason is that a large  $\alpha$  gives high weight to the textual relevance, so that the ranking of the event-partner pairs is affected more by the textual relevance of the events than the together preference. Since events are retrieved in descending order of the textual relevance in the three approaches,

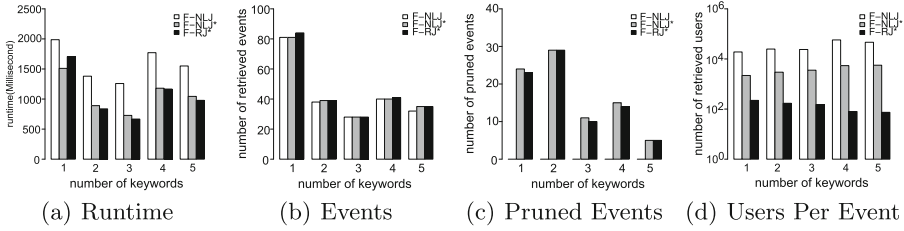


Fig. 4. Varying the number of keywords

the top- $k$  event-partner pairs are determined faster when  $\alpha$  is large. Consistent with the previous results, this experiment also shows the effectiveness of the propose optimizations, i.e, many unpromising events are pruned, and the number of retrieved users per event is reduced.

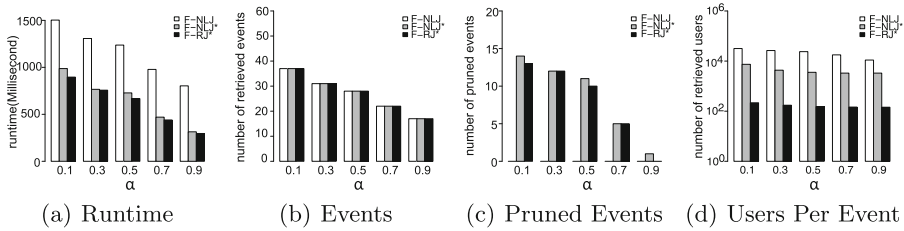


Fig. 5. Varying  $\alpha$

**Summary.** Overall, for a broad range of parameter settings, the proposed optimizations improve the performance of the framework substantially. Unpromising events are pruned. The numbers of users needed for finding partners for events are reduced. In most cases, the ripple join is better than the nested loop join.

## 6 Related Work

**Recommender Systems.** Friend recommendation systems [8] predict user-user relationships (i.e., friendships). They estimates the likelihood that two non-friends will become friends in the future [17,28,31]. Group recommendation [6, 32] explores the preference of a group of users in relation to individual items. In location-based social networks, new locations are recommended to users [29] by taking into account previous user check-ins, the distances of proposed locations to users’ neighborhoods [1], and geographical and social information [16]. In event-based social networks, event recommendation offers a user a set of events by giving consideration to both personal interests and local preferences [30], heterogeneous social relations and implicit feedback [24], social group influences and individual preferences [5], and several contextual signals [18].

All these recommendation techniques only recommend one type of items. The problem studied in this paper adopts a recent recommendation technique [27] that suggests event-partner pairs as a component.

**Rank-Join Algorithms.** Based on the A\* optimization strategy, the  $J^*$  algorithm [21] enables querying of ordered data sets by means of user-defined join predicates. NRA-RJ [9] is a pipelined query operator that produces a global rank from ranked input streams based on a scoring function. Ilyas et al. [10] rank join results progressively during join operations, making use of the individual orders of the inputs.

Ranking (top- $k$ ) queries have also been integrated into relational database systems [11, 13]. Mamoulis et al. [19] identify two phases that any (no random access) NRA algorithm should go through: a growing phase and a shrinking phase. Their LARA algorithm employs a lattice to minimize the computational cost during the shrinking phase. The FRPA rank join operator [4] allows efficient computation of score bounds on unseen join results and prioritizes the I/O requests of the rank join operator based on the potential of each input to generate results with high scores. The Pull/Bound Rank Join (PBRJ) [26] is an algorithm template that generalizes previous rank join algorithms. The idea is to alternate between pulling tuples from input relations and upper bounding the score of join results that use the unread part of the input. The join results collected as tuples are pulled, and the algorithm stops once the top- $k$  buffered results have a score at least equal to the upper bound.

We extend the state-of-the-art rank-join algorithm [10] and propose a framework with optimizations that supports the efficiently processing of  $k$ EP queries.

## 7 Conclusion

This paper introduces the top- $k$  event-partner ( $k$ EP) pair retrieval query that takes the advantages of both event-partner recommendation and keyword-based search. Given a query user, keywords, and a value  $k$ , the query retrieves event-partner pairs from a bipartite event-user graph where events have text descriptions, taking into account both the text relevance of events and the together preference that captures how much the query user prefers to attend an event with a particular partner. For the sake of efficiency, the proposed rank-join based framework comes with three optimizations. The paper’s empirical study offers insight into the proposed techniques, indicating that they are effective and that the framework is practical.

This work opens to a number of promising directions for future work. First, it is worth adapting other existing recommendation techniques developed for events and users to the paper’s setting. Second, it is of interest to consider social relationships between users when processing  $k$ EP queries. Third, it is of interest to understand how the  $k$ EP queries considered can be best processed if the query user’s current location is taken into account.

## References

1. Bao, J., Zheng, Y., Wilkie, D., Mokbel, M.F.: Recommendations in location-based social networks: a survey. *GeoInformatica* **19**(3), 525–565 (2015)
2. Chen, X., Zeng, Y., Cong, G., Qin, S., Xiang, Y., Dai, Y.: On information coverage for location category based point-of-interest recommendation. In: *AAAI*, pp. 37–43 (2015)
3. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.* **66**(4), 614–656 (2003)
4. Finger, J., Polyzotis, N.: Robust and efficient algorithms for rank join evaluation. In: *SIGMOD*, pp. 415–428 (2009)
5. Gao, L., Wu, J., Qiao, Z., Zhou, C., Yang, H., Hu, Y.: Collaborative social group influence for event recommendation. In: *CIKM*, pp. 1941–1944 (2016)
6. Gorla, J., Lathia, N., Robertson, S., Wang, J.: Probabilistic group recommendation via information matching. In: *WWW*, pp. 495–504 (2013)
7. Haas, P.J., Hellerstein, J.M.: Ripple joins for online aggregation. In: *SIGMOD*, pp. 287–298 (1999)
8. Hannon, J., Bennett, M., Smyth, B.: Recommending twitter users to follow using content and collaborative filtering approaches. In: *RecSys*, pp. 199–206 (2010)
9. Ilyas, I.F., Aref, W.G., Elmagarmid, A.K.: Joining ranked inputs in practice. In: *VLDB*, pp. 950–961 (2002)
10. Ilyas, I.F., Aref, W.G., Elmagarmid, A.K.: Supporting top-k join queries in relational databases. *VLDB J.* **13**(3), 207–221 (2004)
11. Ilyas, I.F., Aref, W.G., Elmagarmid, A.K., Elmongui, H.G., Shah, R., Vitter, J.S.: Adaptive rank-aware query optimization in relational databases. *ACM Trans. Database Syst.* **31**(4), 1257–1304 (2006)
12. Ji, X., Qiao, Z., Xu, M., Zhang, P., Zhou, C., Guo, L.: Online event recommendation for event-based social networks. In: *WWW*, pp. 45–46 (2015)
13. Li, C., Chang, K.C., Ilyas, I.F., Song, S.: RankSQL: query algebra and optimization for relational top-k queries. In: *SIGMOD*, pp. 131–142 (2005)
14. Li, H., Ge, Y., Hong, R., Zhu, H.: Point-of-interest recommendations: learning potential check-ins from friends. In: *KDD*, pp. 975–984 (2016)
15. Lian, D., Zhao, C., Xie, X., Sun, G., Chen, E., Rui, Y.: GeoMF: joint geographical modeling and matrix factorization for point-of-interest recommendation. In: *KDD*, pp. 831–840 (2014)
16. Liu, B., Fu, Y., Yao, Z., Xiong, H.: Learning geographical preferences for point-of-interest recommendation. In: *KDD*, pp. 1043–1051 (2013)
17. Lu, Y., Qiao, Z., Zhou, C., Hu, Y., Guo, L.: Location-aware friend recommendation in event-based social networks: a Bayesian latent factor approach. In: *CIKM*, pp. 1957–1960 (2016)
18. Macedo, A.Q., Marinho, L.B., Santos, R.L.: Context-aware event recommendation in event-based social networks. In: *RecSys*, pp. 123–130 (2015)
19. Mamoulis, N., Yiu, M.L., Cheng, K.H., Cheung, D.W.: Efficient top-k aggregation of ranked inputs. *ACM Trans. Database Syst.* **32**(3), 19 (2007)
20. Manotumruksa, J., MacDonald, C., Ounis, I.: Regularising factorised models for venue recommendation using friends and their comments. In: *CIKM*, pp. 1981–1984 (2016)
21. Natsev, A., Chang, Y., Smith, J.R., Li, C., Vitter, J.S.: Supporting incremental join queries on ranked inputs. In: *VLDB*, pp. 281–290 (2001)

22. Ponte, J.M., Croft, W.B.: A language modeling approach to information retrieval. In: SIGIR, pp. 275–281 (1998)
23. Qiao, Z., Zhang, P., Cao, Y., Zhou, C., Guo, L., Fang, B.: Combining heterogenous social and geographical information for event recommendation. In: AAAI, pp. 145–151 (2014)
24. Qiao, Z., Zhang, P., Zhou, C., Cao, Y., Guo, L., Zhang, Y.: Event recommendation in event-based social networks. In: AAAI, pp. 3130–3131 (2014)
25. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. *Commun. ACM* **18**(11), 613–620 (1975)
26. Schnaitter, K., Polyzotis, N.: Optimal algorithms for evaluating rank joins in database systems. *ACM Trans. Database Syst.* **35**(1), 6:1–6:47 (2010)
27. Tu, W., Cheung, D.W., Mamoulis, N., Yang, M., Lu, Z.: Activity-partner recommendation. In: Cao, T., Lim, E.-P., Zhou, Z.-H., Ho, T.-B., Cheung, D., Motoda, H. (eds.) PAKDD 2015. LNCS (LNAI), vol. 9077, pp. 591–604. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-18038-0\\_46](https://doi.org/10.1007/978-3-319-18038-0_46)
28. Wan, S., Lan, Y., Guo, J., Fan, C., Cheng, X.: Informational friend recommendation in social media. In: SIGIR, pp. 1045–1048 (2013)
29. Wang, W., Yin, H., Sadiq, S.W., Chen, L., Xie, M., Zhou, X.: SPORE: a sequential personalized spatial item recommender system. In: ICDE, pp. 954–965 (2016)
30. Yin, H., Sun, Y., Cui, B., Hu, Z., Chen, L.: LCARS: a location-content-aware recommender system. In: KDD, pp. 221–229 (2013)
31. Yu, F., Che, N., Li, Z., Li, K., Jiang, S.: Friend recommendation considering preference coverage in location-based social networks. In: Kim, J., Shim, K., Cao, L., Lee, J.-G., Lin, X., Moon, Y.-S. (eds.) PAKDD 2017. LNCS (LNAI), vol. 10235, pp. 91–105. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-57529-2\\_8](https://doi.org/10.1007/978-3-319-57529-2_8)
32. Yuan, Q., Cong, G., Lin, C.: COM: a generative model for group recommendation. In: KDD, pp. 163–172 (2014)
33. Zhang, W., Wang, J.: A collective Bayesian Poisson factorization model for cold-start local event recommendation. In: KDD, pp. 1455–1464 (2015)



# Local Experts Finding Across Multiple Social Networks

Yuliang Ma<sup>1</sup>(✉), Ye Yuan<sup>1</sup>, Guoren Wang<sup>2</sup>, Yishu Wang<sup>1</sup>, Delong Ma<sup>1</sup>,  
and Pengjie Cui<sup>1</sup>

<sup>1</sup> School of Computer Science and Engineering,  
Northeastern University, Shenyang, China  
ylma.neuer@gmail.com

<sup>2</sup> School of Computer Science and Technology,  
Beijing Institute of Technology, Beijing, China

**Abstract.** The local experts finding, which aims to identify a set of  $k$  people with specialized knowledge around a particular location, has become a hot topic along with the popularity of social networks, such as Twitter, Facebook. Local experts are important for many applications, such as answering local information queries, personalized recommendation. In many real-world applications, complete social information should be collected from multiple social networks, in which people usually participate in and active. However, previous approaches of local experts finding mostly focus on a single social network. In this paper, as far as we know, we are the first to study the local experts finding problem across multiple large social networks. Specifically, we want to identify a set of  $k$  people with the highest score, where the score of a person is a combination of local authority and topic knowledge of the person. To efficiently tackle this problem, we propose a novel framework, KTMSNs (knowledge transfer across multiple social networks). KTMSNs consists of two steps. Firstly, given a person over multiple social networks, we calculate the local authority and the topic knowledge, respectively. We propose a social topology-aware inverted index to speed up the calculation of the two values. Secondly, we propose a skyline-based strategy to combine the two values for obtaining the score of a person. Experimental studies on real social network datasets demonstrate the efficiency and effectiveness of our proposed approach.

**Keywords:** Local experts · Multiple social networks · Multiple graphs

## 1 Introduction

Local expert differs from general topic expert in the sense that local expertise is very limited to a geographical location. Local experts can play an important role in many applications, such as addressing local information queries, social event arrangement [5, 18], and spatial crowdsourcing [19]. For instance, local information queries, like “where can I find the best pub in the Dongcheng District

of Beijing?”, “who is a good dentist in Chinatown of Singapore?”. Indeed, a recent survey by Yahoo! Research finds that people prefer to learning from local experts who know the neighborhood well and have first hand experience [1].

Recently, the *local experts finding* problem has gained increasing attention in social media [6, 14, 15]. Previous works mostly focus on single source social network. However, witnessing the rapid growth of online social networks, people are usually getting involved in multiple social networks simultaneously to enjoy more online social services [12, 25, 26]. For example, people may use Foursquare to share their footprints at different locations or venues with their friends. Meanwhile, they may use Facebook to record something interesting, and turn to Twitter to post comments on the latest news. These social networks sharing common users are formally defined as multiple aligned networks, which is firstly proposed in [12]. Finding local experts from one single source social network may miss some important information, which will cause the inaccuracy of identified experts. For instance, user Alice (or Bob) usually uses Twitter (or Foursquare) to record her (or his) beautiful life, while user Cindy uses both. If we find local experts only in Twitter or Foursquare, Cindy never will be the answer. But Cindy can be the answer if we consider the both social networks, Twitter and Foursquare. Therefore, in this paper, we study the local experts finding problem across multiple heterogeneous social networks.

**Challenges.** Given a social network  $G$ , a local expert is evaluated by a score (*local expertise*) that is related to the information of the local community of  $G$ . The local expertise consists of two values: the local authority, and the topic knowledge. The value of local authority of a person indicates how well does the local community recognize this person. The value of topic knowledge of a person reflects how much does this person know about this topic. Nevertheless, conventional works on a single social network can not be applied to local experts finding over multiple social networks, due to the following challenges:

1. How to evaluate the two types of value, that is, the local authority and the topic knowledge, over multiple heterogeneous social networks.
2. How to fuse the two types of value and make a trade-off for returning a high-accuracy of identified experts.
3. Due to the redundancy of the social information over multiple heterogeneous social networks, how to tackle local experts finding problem efficiently is an urgent requirement.

**Our Contributions.** To address the above challenges in this paper, we propose a new framework, KTMSNs (knowledge transfer across multiple social networks). Two heterogeneous social networks usual have common users, that is, two accounts located on these two social networks correspond to the same person in real life. We call these common users as *anchor users*. The intuition of KTMSNs is that the anchor users can be regard as the “bridges” between two heterogeneous social networks, which can be leveraged to transfer knowledge. Therefore, KTMSNs consists of two main steps.



Firstly, given a person over multiple social networks, we calculate the local authority and the topic knowledge, respectively. We divide the local experts finding problem over multiple social networks (suppose the number of social networks is  $m$ ) into  $(m - 1)$  sub-problems to find local experts across two heterogeneous social networks. For each sub-problem, we propose a social distance based knowledge decay approach to evaluate the local authority and topic authority for all the users across the two heterogeneous social networks.

We also propose a social topology-aware inverted index, STAI, to speed up the calculation of the two values. The STAI index integrates a social distance oracle index and a topic category-aware inverted index. The former is used to calculate the social distance from any user to an anchor user, which can accelerate the social distance based knowledge decay procedure. The latter can be leveraged to achieve high efficiency.

Secondly, we propose a skyline-based strategy to combine the two value for obtaining the score of a person. Usually, someone with a high topic authority may be in a location that is far from the query location, thus the local authority may be relatively low. On the other hand, someone with a high local authority may have a lower topical authority. Skyline-based strategy is suitable to tackle the trade-off obstacle.

Finally, we conduct an extensive experimental evaluation with real datasets to offer insight onto the efficiency of the proposed index and the effectiveness of our proposed algorithms.

The remaining parts of this paper are organized as follows. We formulate our problem in Sect. 2. We give the details of our proposed approaches in Sect. 3. We conduct extensive performance studies and report the results in Sect. 4. We discuss the related work in Sect. 5. Finally, we conclude this paper in Sect. 6.

## 2 Preliminaries and Problem Formulation

In this section, we first describe the terms and notations that we use throughout the paper, and then formally define the local experts finding problem over multiple heterogeneous social networks.

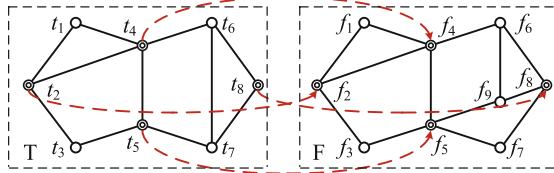
### 2.1 Preliminaries

We model an online social network as an undirected graph  $G = (V, E)$ , where  $V$  is a set of users and  $E$  is a set of social edges. Each vertex  $v \in V$  has a geographical location  $(x, y)$  with longitude  $x$  and latitude  $y$ .

The rise of online social networks (OSNs) and the advances in wireless communication technologies enable Internet users to share life experiences in the physical world via multiple kinds of OSNs. OSNs are mostly heterogeneous, for example, the main element of Instagram is photo, the main element of Foursquare is venue (point-of-interest), whereas short-text is the main element of Twitter. Following the existing work [27], we have some basic concepts with respect to aligned heterogeneous social networks.

**Definition 1.** (Binary Aligned Heterogeneous Social Networks (Bi-AHSNs)). *If two different social networks share some common users, then these two networks are called aligned networks. We name two aligned social networks as binary aligned heterogeneous social networks, which can be formulated as  $\mathbb{G} = ((G_1, G_2), \mathcal{A}_{1,2})$ , where  $G_1$  and  $G_2$  are the two different social networks respectively and  $\mathcal{A}_{1,2} \neq \emptyset$  is the set of undirected anchor links (see Definition 2) between  $G_1$  and  $G_2$ .*

The definition of Bi-AHSNs can be easily extended to multiple aligned heterogeneous social networks (denoted as **Multi-AHSNs**), which can be formulated as  $\mathbb{G} = ((G_1, G_2, \dots, G_m), \mathcal{A}_{1,2}, \mathcal{A}_{1,3}, \dots, \mathcal{A}_{1,m}, \mathcal{A}_{2,3}, \dots, \mathcal{A}_{(m-1),m})$ , where  $G_1, G_2, \dots, G_n$  are different social networks and  $\mathcal{A}_{i,j} \neq \emptyset, i, j \in \{1, 2, \dots, n\}$  is a set of undirected anchor links between  $G_i$  and  $G_j$ .



**Fig. 1.** Example of PAHLEF problem. (Color figure online)

**Definition 2.** (Anchor links). *Let  $V_i$  and  $V_j$  be user sets of  $G_i$  and  $G_j$  respectively. If  $v_i^h \in V_i$  and  $v_j^k \in V_j$  are the accounts of a same user in  $G_i$  and  $G_j$  respectively, we call  $(v_i^h, v_j^k)$  as an undirected anchor link between  $G_i$  and  $G_j$ . The collection of such links is denoted as  $\mathcal{A}_{i,j}$ .*

**Definition 3.** (Anchor users). *User  $v_i^h \in V_i$  is an anchor user in  $G_i$  between  $G_i$  and  $G_j$  iff  $\exists v_j^k \in V_j, (v_i^h, v_j^k) \in \mathcal{A}_{i,j}$ . The set of anchor users in  $G_i$  between  $G_i$  and  $G_j$  is denoted as  $V_i(\mathcal{A}_{i,j}) = \{v_i^h \mid v_i^h \in V_i, \exists v_j^k \in V_j, (v_i^h, v_j^k) \in \mathcal{A}_{i,j}\}$ .*

Following Definition 3, we denote the set of non-anchor users in  $G_i$  between  $G_i$  and  $G_j$  as  $V_i(-\mathcal{A}_{i,j}) = V_i - V_i(\mathcal{A}_{i,j})$ .

**Definition 4.** (Full alignment). *Social networks  $G_i$  and  $G_j$  are fully aligned iff users in  $G_i$  are all anchor users or users in  $G_j$  are all anchor users, that is,  $G_i$  and  $G_j$  are fully aligned iff  $(V_i(\mathcal{A}_{i,j}) = V_i) \vee (V_j(\mathcal{A}_{i,j}) = V_j)$ .*

**Definition 5.** (Partial alignment). *Social networks  $G_i$  and  $G_j$  are partially aligned if there exist users in  $G_i(G_j)$  which are non-anchor users. That is,  $G_i$  and  $G_j$  are partially aligned iff  $(V_i(-\mathcal{A}_{i,j}) \neq \emptyset) \wedge (V_j(-\mathcal{A}_{i,j}) \neq \emptyset) \wedge (\mathcal{A}_{i,j} \neq \emptyset)$ .*

## 2.2 Problem Formulation

As mentioned above, a local expert is evaluated by local expertise, which is related to two kinds of authority values, i.e., local authority and topic knowledge authority. The value of local authority indicates the popularity in the local community. The value of topic knowledge with respect to a person reflects how well dose this person know about a given topic. How to integrate the two values to evaluate one’s local expertise will be discussed in Sect. 3.4. Now we formally define the local experts finding problem over multiple heterogeneous networks.

**Definition 6.** (Local Experts Finding over Partially Aligned Heterogeneous networks, PAHLEF). *Given partially aligned heterogeneous social networks  $\mathbb{G} = ((G_1, G_2, \dots, G_m), \mathcal{A}_{1,2}, \mathcal{A}_{1,3}, \dots, \mathcal{A}_{1,m}, \mathcal{A}_{2,3}, \dots, \mathcal{A}_{(m-1),m})$ , and given a query  $q = \langle t(q), l(q), k \rangle$ , PAHLEF aims to find a set of  $k$  candidates  $\mathbb{C}$  with the highest local expertise with respect to query topic  $t(q)$  and location  $l(q)$ , where  $\mathbb{C} \subseteq (V_1 \cup V_2 \cup \dots \cup V_m)$ .*

*Example 1.* As shown in Fig. 1, there are two different social networks, Twitter ( $T$ ) and Foursquare ( $F$ ), and they are partially aligned. The double loop nodes are the anchor users, the single loop hollow nodes are the non-anchor users, and the red dashed arrows are anchor links. That is,  $\mathcal{A}_{T,F} = \{(t_2, f_2), (t_4, f_4), (t_5, f_5), (t_8, f_8)\}$ , and the set of anchor users in  $T$  between  $T$  and  $F$  is  $V_T(\mathcal{A}_{T,F}) = \{t_2, t_4, t_5, t_8\}$ . Besides, we record each user’s geographical location and activity history in his/her respective social networks. Given a query  $q = \langle music, l(q), 4 \rangle$ , we aim to find a set of 4 users in  $V_T \cup V_F$  with the highest local expertise with respect to *music* and the query location  $l(q)$ .

## 3 Approach

### 3.1 Overview

In this section, we detail our proposed approach, KTMSNs, whose framework is shown in Fig. 2. KTMSNs consists of *data preparation stage*, *social topology aware inverted index construction*, *knowledge decay transfer*, and *trade-off discussion*.

We convert different types of social network data into a unified indication. For location-based platforms (such as Foursquare, Yelp, etc.), common user behavior is check-in at different venues. The venues have their topic categories. We can describe user behavior as topic-category frequency via statistical analysis. We leverage LDA model [3] and TF-IDF based approach [29] to extract topic information. In this paper, we focus on the local experts finding problem over multiple heterogeneous social networks. Based on this intuition, we regard the topic-category frequency description as a part of data preparation. We will not elaborate in detail in the rest of this paper.

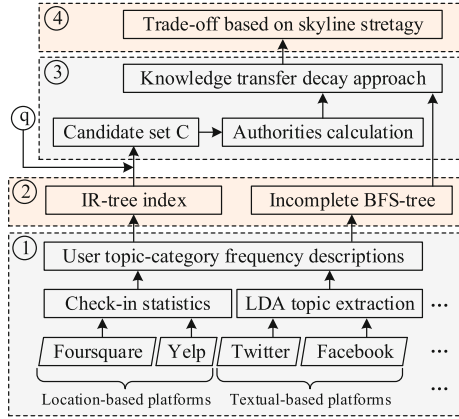


Fig. 2. Framework.

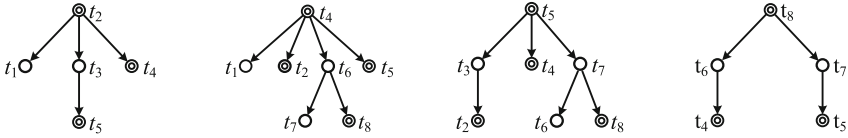
### 3.2 Index Construction

**Social Distance Oracle.** Given partially aligned heterogeneous social networks  $\mathbb{G} = \{G_1, G_2, \dots, G_m, \mathcal{A}_{1,2}, \dots\}$ , we construct a BFS tree for each anchor user in its social network. It is consumptive to build a full BFS tree for each anchor user, since there are more than one anchor user in each social network  $G_i$ . We have Theorem 1 to reduce the scale of the constructed trees.

**Theorem 1.** *Each branch of a BFS-tree rooted at any anchor user can be stopped to do more extensions, if there is another anchor user in this branch.*

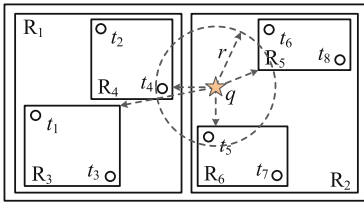
*Proof.* As mentioned in Sect. 1, we utilize an exponential decay model based on social distance to transfer knowledge over multiple social networks. Therefore, we only need to compute the social distance between non-anchor user and anchor user. For any BFS-tree rooted at an anchor user, we should stop to do more extension for this branch, if there is another anchor user in one of the BFS-tree branch. The social distance component of our index consists of all the BFS-tree rooted at all the anchor users in their social networks. For a stopped branch by an anchor user, the rest information can be retrieved from another BFS-tree rooted at that anchor user.

The social distance oracle index comprises incomplete BFS-trees rooted at different anchor users. Thus, we can query the social distance between any non-anchor user  $u$  and any anchor user  $v$  efficiently. We denote the social distance as  $dist(u, v)$ . We first traverse the BFS-tree  $T_v$  rooted at the anchor user  $v$ , if the non-anchor user  $u$  lies in this BFS-tree  $T_v$ , we can get the social distance easily. Otherwise, we get an anchor user set  $A(T_v)$ , in which all the vertices are anchor users appearing in  $T_v$  (except  $v$  itself). We traverse the BFS-trees rooted at the anchor users belonging to  $A(T_v)$ . Once we find the non-anchor user  $u$  at any BFS-tree (suppose this BFS-tree is rooted at an anchor user  $w$ ), we update the social distance between  $u$  and  $v$  as the sum of  $dist(v, w)$  and  $dist(w, u)$ .



(a) BFS-tree rooted at  $t_2$  (b) BFS-tree rooted at  $t_4$  (c) BFS-tree rooted at  $t_5$  (d) BFS-tree rooted at  $t_8$

**Fig. 3.** BFS-tree rooted at anchor users.



(a) User locations

user	topic-category frequency
$t_1$	(food,0.65),(music,0.45)
$t_2$	(sport,0.58),(technology,0.63)
$t_3$	(food,0.85),(sport,0.63)
$t_4$	(sport,0.39),(history,0.72)
$t_5$	(music,0.46),(sport,0.65)
$t_6$	(medicine,0.73),(food,0.69)
$t_7$	(technology,0.76),(music,0.71)
$t_8$	(medicine,0.65),(history,0.51)

(b) User descriptions

**Fig. 4.** Example of topic-category and spatial of social users.

*Example 2.* Take the partially aligned heterogeneous social networks in Fig. 1 as an example. In order to simplify the description, we only give one quick example in terms of Twitter social network. As shown in Fig. 1, there are four anchor users in Twitter social network, i.e.,  $\mathcal{A}_{1,2} = \{t_2, t_4, t_5, t_8\}$ . We construct BFS-tree rooted at each vertex in  $\mathcal{A}_{1,2}$ , which is shown in Fig. 3. The double loop nodes are the anchor users, and the single loop hollow nodes are the non-anchor users.

We take  $t_3$  and  $t_8$  as example to present how to calculate the social distance between a non-anchor user and an anchor user leveraging the social distance oracle. We first traverse the BFS-tree rooted at  $t_8$  to check whether  $t_3$  is in this tree. As a consequence,  $t_3$  is not in  $T(t_8)$ . Thus, we can get the anchor user set  $A(T) = \{t_4, t_5\}$ . Next, we traverse the BFS-tree rooted at the anchor user in  $A(T(t_8))$ . After traversing the BFS-tree rooted at  $t_4$ , the anchor user set  $A(T)$  is updated as  $\{t_5, t_2\}$ . We update the social distance between  $t_3$  and  $t_8$  as  $dist(t_8, t_5) + dist(t_5, t_3) = 2 + 1 = 3$ . The social distance from another social path,  $t_8 \rightarrow t_4 \rightarrow t_2 \rightarrow t_3$ , is large than 3. Finally, the social distance between  $t_3$  and  $t_8$  is 3.

**Topic Category-Aware Inverted Index.** Another component of our index is topic-category-aware inverted index. For each social network platform  $G_i$  in  $\mathbb{G} = \{G_1, G_2, \dots, G_m\}$ , we construct an IR-tree [4, 8, 22] to index the spatial information and the topic-category frequencies with respect to social users.

Topic-category frequency reflects the activity experience of a user, which indicates the knowledge grasped by the user. A topic category hierarchical tree can be extracted from the *Wikipedia*<sup>1</sup>.

The IR-tree is essentially an R-tree [11] extended with inverted files [29]. In this paper, each leaf node in the IR-tree contains entries with a form  $(u, u.\lambda, u.di)$ , where  $u$  refers to a user in the social network,  $u.\lambda$  is the bounding rectangle of  $u$ , and  $u.di$  is an identifier of the description of  $u$ . Each leaf node also contains a pointer to an inverted file with the topic-category knowledge of the users stored in the node.

An inverted file index has two main components, (1) a vocabulary of all distinct topic categories; (2) a posting list for each topic category  $c$  that is a sequence of key-value pairs of those users whose experiences contain  $c$  and corresponding topic category frequency.

Each non-leaf node  $R$  in the IR-tree contains a number of entries of the form  $(cp, cp.\lambda, cp.di)$ , where  $cp$  is the address of a child node of  $R$ ,  $cp.\lambda$  is the *minimum bounding rectangle (MBR)* of all rectangles in entries of the child node, and  $cp.di$  is an identifier of a pseudo text description that is the union of all text descriptions in the entries of the child node. Moreover, the inverted file of the non-leaf node is a summarization of its subtree. Specifically, we record the maximum knowledge score associated with the same topic category in one *MBR*.

Take the *Twitter* social network in Fig. 1 as an example, we suppose that the spatial distribution of users are shown in Fig. 4(a). Figure 4(b) presents the descriptions of users, in which the first column is user list, and the second column includes the topic categories and frequencies.

Figure 5 shows the corresponding IR-tree. Table 1 illustrates the content of the inverted files associated with the nodes. It is worth to note that we construct an IR-tree index for each social platform rather than build an overall IR-tree index. The intuition is to evaluate the local authority and the topical authority of each social user in the all online social networks. It is beneficial to build an overall IR-tree index for the two authorities calculation for anchor users. However, it is not suitable for non-anchor users. Thus, we propose a knowledge decay approach based on social distance to estimate the two authorities for all the users across multiple heterogeneous social networks. An overall IR-tree index is inconsiderable for the knowledge transfer from one social network to another social platform. We will give a detailed elaboration of the knowledge decay method in Sect. 3.3.

### 3.3 Knowledge Decay Across Social Networks

We divide the local experts finding problem over multiple social networks (suppose the number of social networks is  $m$ ) into  $(m - 1)$  sub-problems to find local experts across *two* partially alignment heterogeneous social networks. The authority can be transferred from one social network to another social network

<sup>1</sup> <https://en.wikipedia.org/wiki/Portal:Contents/Portals>.

and merged into a new social network with the total authority score. Therefore, the  $(m - 1)$  sub-problems can be merged into the original problem. Without losing generality, we analyze the details of our proposed approach with respect to the local experts finding problem across *two* partially alignment heterogeneous social networks. We first introduce how to calculate the two authorities value, and then elaborate the authority transfer process between two heterogeneous social networks.

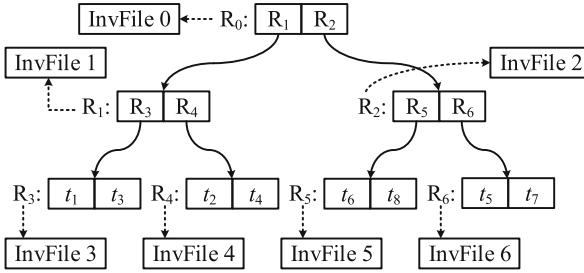


Fig. 5. IR-tree of Fig. 4

Table 1. Partial posting lists of inverted files of the IR-tree

	Food	Music	Sport	Technology
InvFile 0	$(R_1, 0.85), (R_2, 0.69)$	$(R_1, 0.45), (R_2, 0.71)$	$(R_1, 0.63), (R_2, 0.65)$	$(R_1, 0.63), (R_2, 0.76)$
InvFile 1	$(R_3, 0.85)$	$(R_3, 0.45)$	$(R_3, 0.63), (R_4, 0.58)$	$(R_4, 0.63)$
InvFile 2	$(R_5, 0.69)$	$(R_6, 0.71)$	$(R_6, 0.65)$	$(R_6, 0.76)$
InvFile 3	$(t_1, 0.65), (t_3, 0.85)$	$(t_1, 0.45)$	$(t_3, 0.63)$	
InvFile 4			$(t_2, 0.58), (t_4, 0.39)$	$(t_2, 0.63)$
InvFile 5	$(t_6, 0.69)$			
InvFile 6		$(t_5, 0.46), (t_7, 0.71)$	$(t_5, 0.65)$	$(t_7, 0.76)$

**Local Authority.** When a local experts finding query  $q = \langle t(q), l(q), k \rangle$  is proposed, we begin with the focus-based proximity approach to estimate local authority. The intuition is that the more “followers” near location  $l(q)$ , the higher the local authority of a candidate is, which has been demonstrated in [6]. The computation can be formed as:

$$s_l(u, l(q)) = \frac{|\{v_i | v_i \in V(u), d(l(v_i), l(q)) \leq r(l(q))\}|}{|V(u)|}, \tag{1}$$

where  $s_l(u, l(q))$  denotes the local authority score of a social user  $u$ ,  $v_i$  is one of the “followers”  $V(u)$  associated with  $u$ ,  $d(l(v_i), l(q))$  represents the spatial distance between  $v_i$  and  $l(q)$ , and  $r(l(q))$  indicates a radius around the query location  $l(q)$ .

We utilize R-tree structure to accelerate the local authority preprocessing. As shown in Fig. 4(a), the star is query location, and the gray dashed circle is the region constrained by radius  $r(l(q))$ . The lower bound of spatial distance with respect to a user in a *MBR* can be easily obtained by computing the distance from  $l(q)$  to the bounding rectangle. The distance of each user in a *MBR* to  $l(q)$  is larger than  $r(l(q))$ , if the lower bound is larger than  $r(l(q))$ . Thus, the users lying in such *MBR* can not contribute the count of numerator in Eq. 1 without any distance computation.

**Topical Authority.** In order to evaluate the topical authority, we first leverage the topic category-aware inverted index to generate a candidate set  $C$  by mapping the query topic  $t(q)$  to the topic category hierarchical tree. If the mapped node in the tree is a non-leaf node, we regard the sub-tree of this non-leaf node as the query topic extension. For instance,  $t(q) = \{food\}$  is non-leaf in the topic category hierarchical tree, we extend the query topic to  $t^*(q) = \{food, Chinese\ food, Indian\ restaurant, \dots\}$ , in which each element is one node of the sub-tree. We retrieve these elements from the vocabulary of the inverted index, each user with non-vanishing experience of any elements in  $t^*(q)$  is contained into the candidate set  $C$ . Whereafter, we need to calculate the authorities of anchor users in the two social networks respectively. Consequently, we put the anchor users into the candidate set  $C$ , since all of them are required to calculate the two authorities score.

User topical authority  $Ta$  and the activity experience  $\mathcal{E}$  have a mutual reinforcement relationship. A user with high authority in a topic area would be active in the topic area. Activities of a topic area would enhance the corresponding topical authority. More specifically, a user’s topical authority can be calculated by integrating the activity experiences generated by the user. The activity experiences associated with a topic area can be counted via statistic topic category frequencies.

Given an activity record (i.e., user posting history, check-in history, etc.), we can build an adjacent matrix  $R$  between user and topic category. In this matrix, an item  $r_{ij}$  represents the topic category frequency,  $0 \leq i < |U|$ ,  $0 \leq j < |T|$ . For instance, the matrix specified by Fig. 4(b) can be represented as follows (suppose the topic category set denoted as  $T = \{food, music, sport, technology, history, medicine\} = \{t_0, t_1, \dots, t_5\}$ ).

$$R = \begin{matrix} & r_0 & r_1 & r_2 & r_3 & r_4 & r_5 \\ \begin{matrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{matrix} & \left[ \begin{array}{cccccc} 0.65 & 0.45 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.58 & 0.63 & 0 & 0 \\ 0.85 & 0 & 0.63 & 0 & 0 & 0 \\ 0 & 0 & 0.39 & 0 & 0.72 & 0 \\ 0 & 0.46 & 0.65 & 0 & 0 & 0 \\ 0.69 & 0 & 0 & 0 & 0 & 0.73 \\ 0 & 0.71 & 0 & 0.73 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.51 & 0.65 \end{array} \right] \end{matrix}$$



Therefore, the mutual reinforcement relationship of user topical authority  $Ta = (ta_0, ta_1, \dots, ta_m)$  and activity experience  $\mathcal{E} = (e_0, e_1, \dots, e_n)$  are represented as follows:

$$ta_i = \sum_{u_j \in U} r_{ij} \times e_j, \tag{2}$$

$$e_j = \sum_{r_i \in R} r_{ji} \times ta_i, \tag{3}$$

where  $ta_i$  denotes the authority of topic category  $ta_i$ , and  $e_j$  indicates the activity experience of user  $u_j$ . Thus, the matrix form can be represented

$$Ta = R \cdot E, \tag{4}$$

$$E = R^T \cdot Ta, \tag{5}$$

If we use  $Ta_k$  and  $E_k$  to denote topic authority and activity experience at the  $k$ th iteration, the iterative processes for generating the topical authority results are

$$Ta_k = R^T \cdot R \cdot E_{k-1}, \tag{6}$$

$$E_k = R \cdot R^T \cdot Ta_{k-1}, \tag{7}$$

Starting with  $Ta_0 = E_0 = (1, 1, \dots, 1)$ , we can evaluate the topical authority via the power iteration method.

**Authorities Transfer.** After the two authorities can be calculated efficiently over a single social network, we are ready to evaluate the two types of authorities with respect to each candidate  $c \in C$  over Bi-AHSNs. If  $c$  is an anchor user, we sum the two types of authorities respectively. In contrast, if  $c$  is a non-anchor user, we propose an exponential decay model based on social distance to estimate the two kinds of authorities.

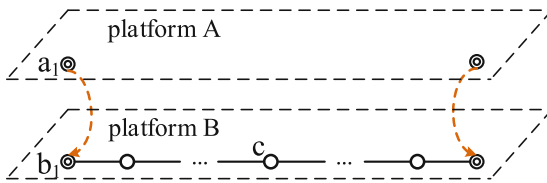


Fig. 6. Knowledge transfer via anchor users

The main idea of the exponential decay model is that knowledge can be transferred from one social platform to another social platform through anchor links. As a consequence, the anchor users are the knowledge transfer intermediaries. As shown in Fig. 6, there are two social platforms A and B. In order to evaluate the authorities of a non-anchor candidate user (suppose  $c$ ) over two partially aligned heterogeneous social networks, we first find the shortest social distance between  $c$  and all anchor users in the platform where  $c$  lies in, which

can be obtained via the BFS-tree index presented in Subsect. 3.2. Assume the anchor user set between two social platforms is denoted as  $\mathcal{A}_{AB}$  (abbreviated as  $\mathcal{A}$ ). Thus, the authorities score can be estimated as follow:

$$S_l(c, l(q)) = s_{lB}(c, l(q)) + \sum_{a_i \in \mathcal{A}} 2^{-dist(a_i, c)} \cdot s_{lA}(a_i, l(q)), \quad (8)$$

where  $S_l(c, l(q))$  represents the local authority of non-anchor candidate user  $c$  belongs to platform  $A$  over the two partially aligned heterogeneous social platforms,  $s_{lB}(c, l(q))$  denotes local authority of  $c$  over one single social platform  $B$ , and  $dist(a_i, c)$  indicates the shortest social distance between an anchor user  $a_i \in \mathcal{A}$  and  $c$ . The topical authority is transferred similarly to the local authority.

### 3.4 Trade-Off Discussion

Albeit we can evaluate the two types of authorities over multiple partially aligned heterogeneous networks, there is still an obstacle. In order to satisfy the diversity of query results, we need to return a collection of  $k$  candidates with the highest local expertise associated with query topic  $t(q)$  and location  $l(q)$ . How to make a trade-off the topical authority and the local authority is a challenge. In this paper, we regard the two authorities as two dimensions of local experts, and the well-known *skyline* [2, 17] strategy is suitable tackle two dimensions trade-off problem.

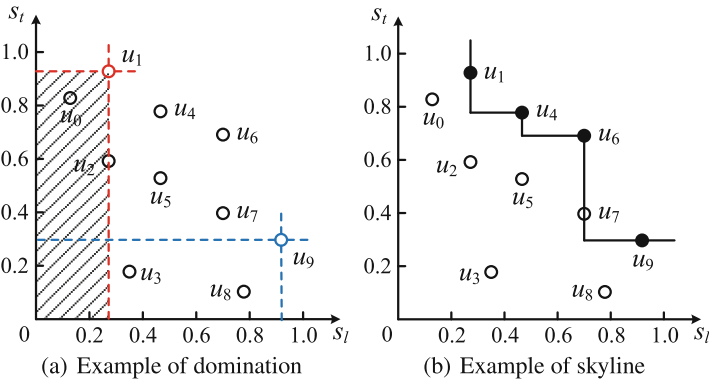


Fig. 7. Example of two authorities trade-off (Color figure online)

The skyline strategy is important for several applications involving multi-criteria decision making. Given a set of objects  $\{o_1, o_2, \dots, o_N\}$ , the strategy returns  $k$  objects as the answer of top- $k$  query such that any object  $o_i$  in the returned result is not dominated by another object  $o_j$ . In this paper, the multi-criteria are the topical authority and the local authority. We have the definition of domination as follow.

**Definition 7** (Domination). *Given a set of users  $\{u_0, u_1, \dots, u_m\}$ , we regard the two types of authorities as two orthogonal dimensions. A user  $u_i$  in the two-dimensional space can dominate the users lying in the lower-left subspace. And we denote such a subspace as the domination region of  $u_i$ .*

The domination number of user  $u_i$  is the number of users lying in  $u_i$ 's domination region. As a consequence, we rank the users in the candidate set  $C$  by the descending order of domination number. Finally, the top- $k$  users are returned as answer of the local experts finding problem.

*Example 3.* Take Fig. 7 as an example, suppose there are eight users in a candidate set  $C$ , the  $s_l$ -axis demotes the local authority score, and the  $s_t$ -axis denotes the topical authority score. As shown in Fig. 7(a), the rectangular region formed by the red dot line and the two axes is the domination region of  $u_1$ . The users  $u_0$  and  $u_2$  are both dominated by  $u_1$ , since they are lying in the domination region of  $u_1$ . The domination number of  $u_1$  is two. If  $k = 4$ , we can obtain a skyline shown in Fig. 7(b) by ranking the users' domination number.

## 4 Experiments

### 4.1 Experimental Setting

**Datasets.** We use two real-world datasets in our experiments.

- Foursquare. We crawled the Foursquare datasets via Foursquare API<sup>2</sup> from Nov. 2014 to Jan. 2016, which is comprised of 76,503 users and 1,531,357 social ties. For each user, we collected the check-ins in this period of time and the geographical location. Each check-in is a pair of user and venue (i.e., point-of-interest). Totally, we crawled 299,995 venues located in Singapore and 969,549 check-ins. Each venue has its category type description, such as museum, Chinese Restaurant.
- Twitter. We gathered the Twitter datasets via Twitter API<sup>3</sup> from Nov. 2014 to Jan. 2016, which contains 160,338 users and 2,405,628 social ties. For each user, we collected a set of the most recent ( $\leq 1000$ ) user-generated tweets and the geographical location.

There are 15,281 common users in the two datasets, that is, these users have both Foursquare account and Twitter account, and the corresponding account pair has already been matched by some strong evidence, like that they are registered via a same email. Meanwhile, we utilize the method depicted in the data preparation stage to discovery users' topic-category frequencies. A simple example is described in Fig. 4(b).

**Queries.** As our query model is defined as triple  $q = \langle t(q), l(q), k \rangle$ , we randomly choose 50 query keyword (belongs to topic category hierarchical tree) and

<sup>2</sup> <https://developer.foursquare.com/>.

<sup>3</sup> <https://dev.twitter.com/>.

query location (lying at Singapore) pairs, and set  $k = 10$  as default value to evaluate the algorithm performance. These query combinations consist of 20 general topic queries like “technology” and 30 specific topic queries like “Sichuan cuisine foodies”. We allocated the query location set as {the West of Singapore, the East of Singapore, the South of Singapore, the North of Singapore, the Whole Singapore}, which means we have *four* general query topics and *six* specific query topics under each general query topic.

**Compared Algorithms.** Since we are the first work to tackle the local experts finding problem over multiple partially aligned heterogeneous social networks, we take our proposed method, KTMSNs, with one single social network as the baseline algorithm. We implement our algorithm without the knowledge transfer procedure over Twitter datasets. Consequentially, the comparison algorithms implemented in this paper are elaborated as follows:

**Naive:** We utilized the state-of-the-art local experts finding algorithm proposed in [20] on Twitter uniquely as the baseline algorithm. Obviously, the naive solution may reduce the accuracy of identified experts. The naive solution separates the multiple networks to evaluate local expertise, which will miss some important information. People are usually getting involved in multiple social networks simultaneously.

**LEF+T:** We implemented our proposed approach on Twitter datasets without the knowledge transfer procedure.

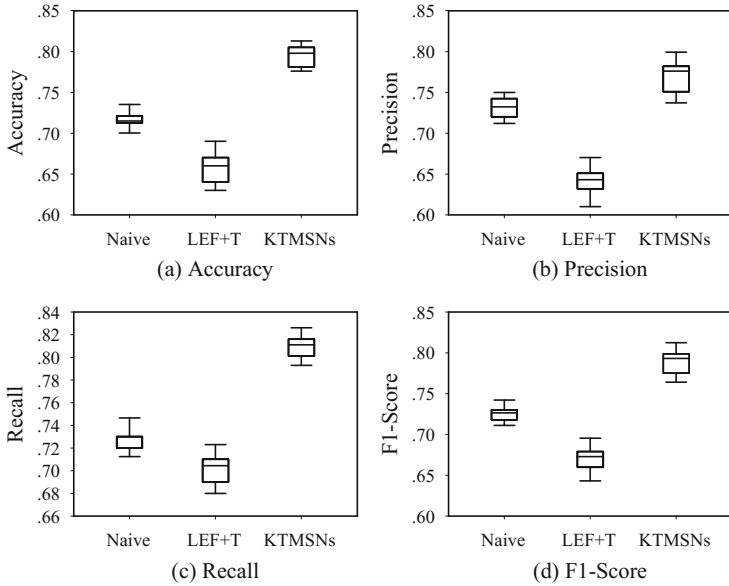
**KTMSNs:** Our proposed approach implemented over the partially aligned multiple social networks, i.e, the partially aligned Twitter and Foursquare datasets.

Besides, we implement another version of KTMSNs algorithm without the index, denoted as **KTMSNs-Index**, to evaluate the performance of our constructed index.

## 4.2 Experimental Results and Analysis

**Performance Comparison.** In this set of experiments, we study the performance with respect to accuracy, precision, recall and F1-Score over the above three algorithms (Naive, LEF+T, and KTMSNs). The results are shown in Fig. 8.

From Fig. 8, we can clearly see that KTMSNs algorithm consistently outperforms the baselines in local experts finding. The results on different metrics shows that the performance improvement of KTMSNs over baselines is about 10%–15%. This observation validates that it is significant to consider heterogeneous networks for finding local experts. As shown in Fig. 8(a), the accuracy of local experts finding by KTMSNs algorithm is about 15% higher than the results of baselines. The reason is that people are usually activity on multiple different social networks simultaneously. Finding local experts over a single social network will miss some important information, which will reduce the accuracy. Figures 8(b), (c) and (d) show the results of precision, recall and F1-score, respectively. The reason of the results of the other metrics is analogical.



**Fig. 8.** Performance comparison

From the results shown in Fig. 8, we can find that the performance of the Naive algorithm outperforms the LEF+T approach. The reason is that Naive is implemented over two social networks. In contrast, LEF+T only leverages the information obtained from one single social network, i.e., Twitter.

**Efficiency Evaluation.** In this set of experiments, we study the efficiency of different algorithms to find local experts on real datasets. Figure 9 shows the comparison on overall running time of LEF+T, LEF+F (our proposed approach implemented on Foursquare datasets), KTMSNs-Index, and KTMSNs. Although the baselines run faster than KTMSNs algorithm, the efficiency of KTMSNs is accredited.

As shown in Fig. 9, we can see that the runtime of KTMSNs algorithm is far less than that of KTMSNs-Index algorithm, which reflects that our proposed index is can largely improve the query efficiency.

**The Effect of  $k$ .** In this set of experiments, we study the impact of query parameter  $k$  on the accuracy and the overall running time of KTMSNs algorithm.

Figure 9 shows the comparison on overall runtime of LEF+T, LEF+F, KTMSNs-Index, and KTMSNs for varying query parameter  $k$ . These algorithms run slower with  $k$  increasing. The runtime of all algorithms increase slow, since all the three algorithms have index mechanism to accelerate query processing.

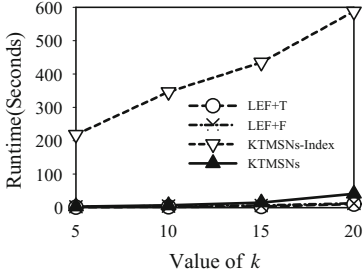


Fig. 9. Runtime vs  $k$

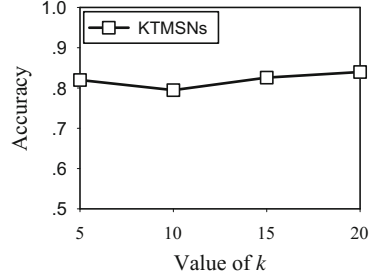


Fig. 10. Accuracy vs  $k$

Figure 10 shows the accuracy of KTMSNs algorithm for varying query parameter  $k$ . We can see that KTMSNs can generate results with high quality and effectiveness with  $k$  increasing.

## 5 Related Work

Experts finding [9, 28], which aims at identifying people with the relevant expertise on a given query topic, has attracted intensive attention in past years [7, 13, 21]. Previously, most work focuses on expert finding problem with considering spatial information. For example, Pal and Counts [16] proposed a probabilistic clustering framework-based approach to identify authority experts in a query topic using both nodal and topical features. Ghosh et al. [10] built the Cognos system, which leveraged Twitter lists to identify the user’s expertise, and they reported that their system works as well as Twitter’s official system to identify top users in a query topic. A further study [20] utilized multiple types of relations in Twitter to identify experts with respect to a query topic.

Local experts finding problem aims at search experts with the highest local expertise in query topic and a given query location, which differs from the traditional expert finding problem. Cheng et al. [6] proposed a geo-spatial-driven approach for identifying local experts that leveraged the fine-grained GPS coordinates of Twitter users. Besides, they gave a local expertise framework that integrated both users’ topical expertise and users’ local authority. Li et al. [14] proposed a range of probabilistic models of local expertise based on geo-tagged social network streams. They assume that frequent visits result in greater familiarity with the location in question and exploited the spatio-temporal information from users’ online check-in profiles. Niu et al. [15] explored a geo-spatial learning-to-rank framework for identifying local experts. While those works tackling local experts finding problem focused on one single platform, which differs from our problem. We are the first work to study local experts finding problem over multiple partially aligned social networks.

In the context of multiple partially align social networks, previous works mostly focused on anchor links prediction [12, 23], which aimed at finding that

different accounts registered on different platforms belong to the same natural person. Another kind of research topic over multiple partially aligned social networks is user link prediction leveraging transfer learning method [25, 27]. Besides, there are some other kinds of previous works over multiple partially aligned social networks, such as [24].

## 6 Conclusion

In this paper, we studied a novel problem of local experts finding in multiple partially aligned heterogeneous social networks. We precisely defined this problem as partially aligned heterogeneous networks local experts finding (PAHLEF for short). In order to tackle the PAHLEF problem, we proposed a novel model, knowledge transfer across multiple social networks (KTMSNs for short). Firstly, we constructed a social topology aware inverted index to accelerate query processing. Then, we divided the local experts finding problem over multiple social networks into some sub-problems. For each sub-problem, we proposed a social distance based knowledge decay approach to evaluate local authority and topic authority. Finally, a skyline-based strategy is elaborated for trade-off with respect to the two authority criteria. Experiment results show that our algorithm has high efficiency and effectiveness to find local experts in multiple partially aligned heterogeneous networks.

**Acknowledgments.** This research is partially funded by the National Key Research and Development Program of China (Grant No. 2016YFC1401900), the National Natural Science Foundation of China (Grant Nos. 61572119, 61572121, 61622202, 61732003, 61729201, 61702086, and U1401256), the Fundamental Research Funds for the Central Universities (Grant Nos. N171604007, and N171904007), the Natural Science Foundation of Liaoning Province (Grant no. 20170520164), and the China Postdoctoral Science Foundation (Grant no. 2018M631806).

## References

1. Antin, J., de Sa, M., Churchill, E.F.: Local experts and online review sites. In: Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work Companion, pp. 55–58. ACM (2012)
2. Bai, M., Xin, J., Wang, G.: Subspace global skyline query processing. In: Proceedings of the 16th International Conference on Extending Database Technology, pp. 418–429. ACM (2013)
3. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. *J. Mach. Learn. Res.* **3**, 993–1022 (2003)
4. Cao, X., Cong, G., Guo, T., Jensen, C.S., Ooi, B.C.: Efficient processing of spatial group keyword queries. *ACM Trans. Database Syst. (TODS)* **40**(2), 13 (2015)
5. Cheng, Y., Yuan, Y., Chen, L., Giraud-Carrier, C., Wang, G.: Complex event-participant planning and its incremental variant. In: IEEE International Conference on Data Engineering, pp. 859–870 (2017)

6. Cheng, Z., Caverlee, J., Barthwal, H., Bachani, V.: Who is the barbecue king of Texas?: A geo-spatial approach to finding local experts on twitter. In: Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 335–344. ACM (2014)
7. Chi, E.H.: Who knows?: Searching for expertise on the social web: technical perspective. *Commun. ACM* **55**(4), 110 (2012)
8. Cong, G., Jensen, C.S., Wu, D.: Efficient retrieval of the top-k most relevant spatial web objects. *Proc. VLDB Endow.* **2**(1), 337–348 (2009)
9. Fang, Y., Si, L., Mathur, A.P.: Discriminative models of integrating document evidence and document-candidate associations for expert search. In: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 683–690. ACM (2010)
10. Ghosh, S., Sharma, N., Benevenuto, F., Ganguly, N., Gummadi, K.: Cognos: crowdsourcing search for topic experts in microblogs. In: Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 575–590. ACM (2012)
11. Guttman, A.: R-trees: a dynamic index structure for spatial searching, vol. 14. ACM (1984)
12. Kong, X., Zhang, J., Yu, P.S.: Inferring anchor links across multiple heterogeneous social networks. In: Proceedings of the 22nd ACM International Conference on Information and Knowledge Management, pp. 179–188. ACM (2013)
13. Lappas, T., Liu, K., Terzi, E.: A survey of algorithms and systems for expert location in social networks. In: Aggarwal, C. (ed.) *Social Network Data Analytics*, pp. 215–241. Springer, Boston (2011). [https://doi.org/10.1007/978-1-4419-8462-3\\_8](https://doi.org/10.1007/978-1-4419-8462-3_8)
14. Li, W., Eickhoff, C., de Vries, A.P.: Probabilistic local expert retrieval. In: Ferro, N., et al. (eds.) *ECIR 2016. LNCS*, vol. 9626, pp. 227–239. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-30671-1\\_17](https://doi.org/10.1007/978-3-319-30671-1_17)
15. Niu, W., Liu, Z., Caverlee, J.: On local expert discovery via geo-located crowds, queries, and candidates. *ACM Trans. Spat. Algorithms Syst. (TSAS)* **2**(4), 14 (2016)
16. Pal, A., Counts, S.: Identifying topical authorities in microblogs. In: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, pp. 45–54. ACM (2011)
17. Papadias, D., Tao, Y., Fu, G., Seeger, B.: An optimal and progressive algorithm for skyline queries. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, pp. 467–478. ACM (2003)
18. She, J., Tong, Y., Chen, L., Cao, C.C.: Conflict-aware event-participant arrangement and its variant for online setting. *IEEE Trans. Knowl. Data Eng.* **28**(9), 2281–2295 (2016)
19. Tong, Y., Chen, L., Zhou, Z., Jagadish, H.V., Shou, L., Lv, W.: SLADE: a smart large-scale task decomposer in crowdsourcing. *IEEE Trans. Knowl. Data Eng.* **30**(8), 1588–1601 (2018)
20. Wei, W., Cong, G., Miao, C., Zhu, F., Li, G.: Learning to find topic experts in twitter via different relations. *IEEE Trans. Knowl. Data Eng.* **28**(7), 1764–1778 (2016)
21. Weng, J., Lim, E.P., Jiang, J., He, Q.: TwitterRank: finding topic-sensitive influential Twitterers. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining, pp. 261–270. ACM (2010)
22. Wu, D., Cong, G., Jensen, C.S.: A framework for efficient spatial web object retrieval. *Int. J. Very Large Data Bases (VLDB)* **21**(6), 797–822 (2012)



23. Zafarani, R., Tang, L., Liu, H.: User identification across social media. *ACM Trans. Knowl. Discov. Data (TKDD)* **10**(2), 16 (2015)
24. Zhan, Q., Zhang, J., Wang, S., Yu, P.S., Xie, J.: Influence maximization across partially aligned heterogeneous social networks. In: Cao, T., Lim, E.-P., Zhou, Z.-H., Ho, T.-B., Cheung, D., Motoda, H. (eds.) *PAKDD 2015, Part I. LNCS (LNAI)*, vol. 9077, pp. 58–69. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-18038-0\\_5](https://doi.org/10.1007/978-3-319-18038-0_5)
25. Zhang, J., Kong, X., Philip, S.Y.: Predicting social links for new users across aligned heterogeneous social networks. In: *2013 IEEE 13th International Conference on Data Mining (ICDM)*, pp. 1289–1294. IEEE (2013)
26. Zhang, J., Kong, X., Yu, P.S.: Transferring heterogeneous links across location-based social networks. In: *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, pp. 303–312. ACM (2014)
27. Zhang, J., Yu, P.S., Zhou, Z.H.: Meta-path based multi-network collective link prediction. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1286–1295. ACM (2014)
28. Zhang, J., Tang, J., Li, J.: Expert finding in a social network. In: Kotagiri, R., Krishna, P.R., Mohania, M., Nantajeewarawat, E. (eds.) *DASFAA 2007. LNCS*, vol. 4443, pp. 1066–1069. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-71703-4\\_106](https://doi.org/10.1007/978-3-540-71703-4_106)
29. Zobel, J., Moffat, A.: Inverted files for text search engines. *ACM Comput. Surv. (CSUR)* **38**(2), 6 (2006)



# SBRNE: An Improved Unified Framework for Social and Behavior Recommendations with Network Embedding

Weizhong Zhao<sup>1,2</sup>(✉), Huifang Ma<sup>3</sup>, Zhixin Li<sup>4</sup>, Xiang Ao<sup>5,7</sup>, and Ning Li<sup>6</sup>

<sup>1</sup> School of Computer, Central China Normal University, Wuhan, China  
zhaowz81@163.com

<sup>2</sup> Hubei Key Laboratory of Artificial Intelligence and Smart Learning,  
Central China Normal University, Wuhan, China

<sup>3</sup> College of Computer Science and Engineering,  
Northwest Normal University, Lanzhou, China

<sup>4</sup> Guangxi Key Lab of Multi-source Information Mining and Security,  
Guangxi Normal University, Guilin, China

<sup>5</sup> Key Laboratory of Intelligent Information Processing,  
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China

<sup>6</sup> Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

<sup>7</sup> University of Chinese Academy of Sciences, Beijing, China

**Abstract.** Recent years have witnessed the fast growing and ubiquity of social media which has significantly changed the social manner and information sharing in our daily life. Given a user, social (*i.e.* friend) recommendation and behavior (*i.e.* item) recommendation are two types of popular services in social media applications. Despite the extensive studies, few existing work has addressed both tasks elegantly and effectively. In this paper, we propose an improved unified framework for Social and Behavior Recommendations with Network Embedding (SBRNE for short). With modeling social and behavior information simultaneously, SBRNE integrates social recommendation and behavior recommendation into a unified framework. By employing users' latent interests as a bridge, social and behavior information is modeled effectively to improve performance of social and behavior recommendations all together. In addition, an efficient network embedding procedure is introduced as a pre-training step for users' latent representations to improve effectiveness and efficiency of recommendation tasks. Extensive experiments on real-world datasets demonstrate the effectiveness of SBRNE.

**Keywords:** Social recommendation · Behavior recommendation · Network embedding · Probabilistic matrix factorization

## 1 Introduction

With the rapid development of information technology, the amount of information available on social media becomes larger and larger. Effective tools are in great demand to help users for discovering their interested information. To meet the increasing demand, many recommender systems have emerged, *e.g.* recommender systems for movies, books, friends, collaborators, etc.

Generally, there are two types of popular recommendation tasks: social (user) recommendation and behavior (item) recommendation. The goal of social recommendation (*a.k.a.* link prediction) [3, 10, 29] is to derive a list of social links or friends for users, with whom most probably users would like to construct associations, while the goal of behavior recommendation [4, 6–9, 15, 17, 30], is to derive a list of items, on which users might conduct an appropriate behavior (*e.g.* rate or purchase). To date, either social or item recommendation is extensively explored by current research according to different strategies [12–14, 16, 18, 19, 21]. However, exploiting social and behavior information simultaneously to improve the performance of recommendations is still in an urgent demand.

Although there are several approaches that can model social and behavior information contemporaneously, we argue that they suffer from, still, the following shortcomings. The primary one is that behavior information is usually regarded as a fixed auxiliary source for improving social recommendation and vice versa. For instance, Wang et al. [24] adopted the behavior associations between users and items as a kind of auxiliary information for link prediction on social network. Liu et al. [14] learned the dependencies between users to facilitate item recommendations. Among these methods, the auxiliary modality is usually predefined and remains unchanged during the learning processing. However, social and behavior information could make both social and behavior recommendations enhance their performance mutually because of their coherent correlations. Hence it is desirable for an effective and efficient unified framework to simultaneously perform social and behavior recommendation by exploiting the clear duality between them. Second, most approaches [1, 16, 18, 19, 22, 25, 26] models users' latent profiles with more than one random variables, providing more expressiveness at the price of introducing more noise which may lead to undesirable recommendation performance.

Network embedding is a family of network representation methods which learn distributed low-dimensional vector representations for vertices with preserving the sparse structural information of the network. The representative approaches include DeepWalk [20], LINE [23]. Network embedding is typically used as a preprocessing/pre-training step to boost the performance of subsequent network analysis tasks [2], such as vertex classification, vertex clustering, network visualization and link prediction.

Motivated by above observations, in this paper, we propose a framework called SBRNE, which models both social and behavior information simultaneously via a probabilistic matrix factorization based approach. More specifically, by utilizing users' profiles as a bridge, social and behavior information is employed to make social and behavior recommendations to enhance their

performance mutually. In addition, a simple but effective network embedding method is designed as a pre-training procedure for users' profiles to improve the performance of recommendation tasks. With SBRNE in hand, two tasks of social and behavior recommendations are addressed effectively under a unified model. Generally, our contributions are summarized as follows.

- We propose a unified probabilistic matrix factorization based framework SBRNE for social and behavior recommendation. With the single framework, performance of social recommendation and behavior recommendation can be improved simultaneously.
- An effective and efficient network embedding approach is proposed as a pre-training procedure for users' latent variables to boost the performance of the proposed framework on recommendation tasks.
- Comprehensive experiments are conducted on three real-world datasets, and evaluation results show that SBRNE performs better than the state-of-the-art approaches.

Note that in the whole paper, we use interchangeably social recommendation and user recommendation, behavior recommendation and item recommendation, respectively.

## 2 SBRNE: An Improved Unified Framework for Social and Behavior Recommendations with Network Embedding

Assume we have  $N$  users  $\mathbb{U} = \{u_1, \dots, u_N\}$  and  $M$  items  $\mathbb{V} = \{v_1, \dots, v_M\}$ . The input of SBRNE consists of social associations between users  $S \in \mathbb{R}^{N \times N}$ , behavior information between users and items  $R \in \mathbb{R}^{N \times M}$ .  $S_{ik} = 1$  if  $u_i$  has a social connection with  $u_k$ ; otherwise  $S_{ik} = 0$ .  $R$  may have different meanings under different scenarios. For example,  $R_{ij}$  denotes the rating of user  $u_i$  on the item  $v_j$ , if the item is a movie or other product. If the item is a tag of users,  $R_{ij} = 1$  denotes user  $u_i$  has the tag  $v_j$ ; otherwise  $R_{ij} = 0$ .

To formulate the framework conveniently, we assume the social associations  $S$  is a symmetric matrix, and  $R$  is a matrix of ratings made by users on items. More specifically,  $S_{ik} \in \{0, 1\}$ , and  $R_{ij} \in \{1, 2, 3, 4, 5\}$ . Note however that all of the following derivations can be readily generalized to other situations.

### 2.1 Network Embedding as Pre-training for Users' Latent Profiles

Before training the whole framework, we propose a network embedding approach as a pre-training procedure for users' latent variables  $U \in \mathbb{R}^{D \times N}$  where  $D$  is the dimension of representations. Given the social network  $S \in \mathbb{R}^{N \times N}$ , for each edge  $S_{ik}$ , the structural similarity (SS) [27] between users  $i$  and  $k$  is calculated as follows:

$$SS_{ik} = \frac{|I(i) \cap I(k)|}{\sqrt{|I(i)| \cdot |I(k)|}} \quad (1)$$

where for each vertex  $i$ ,  $\Gamma(i) = \{v \in V | S_{iv} = 1\} \cup \{i\}$ . In the right-hand side of Eq. (1), the numerator is the number of shared neighbors (including themselves) of users  $i$  and  $k$ , and the denominator is the normalization item. From Eq. (1), it's known that the definition of structure similarity takes into account implicitly not only the first-order proximity (vertices adjacent to each edge) but also the second-order proximity (shared neighbors) between pairs of vertices.

Given each edge  $S_{ik}$ , the compatibility between vertices  $i$  and  $k$  is defined by a sigmoid function as follows:

$$C_{ik} = \frac{1}{1 + \exp(-U_i^T \cdot U_k)} \quad (2)$$

where  $U_i \in R^D$  is the low-dimensional vector representation of vertex  $i$ . To preserve the structure information in the social network, the optimization objective of our network embedding method is defined as follows:

$$O = \frac{1}{2} \sum_{(i,k) \in E} (C_{ik} - SS_{ik})^2 + \frac{1}{2} \rho \sum_{i=1}^N \|U_i\|_2^2 \quad (3)$$

in which the first term in the right-hand side is the squared error between the compatibility and structure similarity of all edges in the network, the second term is the  $l_2$ -norm regularizer multiply a trade-off coefficient  $\rho$ . The gradient of the objective *w.r.t.*  $U_i$  is derived readily as:

$$\frac{\partial O}{\partial U_i} = \sum_{j \in N(i)} ((C_{ij} - SS_{ij}) \cdot C_{ij} \cdot (1 - C_{ij}) \cdot U_j) + \rho \cdot U_i \quad (4)$$

where  $N(i)$  is the set of neighbors of user  $i$ .

To minimize the objective in Eq. (3), a coordinate gradient descent approach is applied due to the coupling of latent profiles of different vertices in Eq. (4). Specifically, for each user  $i \in V$ , the low-dimensional representation is updated alternatively by:

$$U_i = U_i^{old} - \tau \cdot \frac{\partial O}{\partial U_i} \quad (5)$$

where  $\tau$  is a learning rate.

Note that the reason we choose the reconstruction error term and regularization term in Eq. (3) for network embedding is just the efficiency, since the network embedding is only utilized as a pre-training (*i.e.* initialization) procedure for users' latent profiles before optimizing SBRNE. The experimental results demonstrate that the network embedding procedure is effective enough for advancing the recommendation performance.

## 2.2 SBRNE Framework

The fundamental idea of SBRNE is based on the commonly accepted assumption that users have certain latent interests and items have certain latent features.

Moreover, network embedding can be used effectively as a low-dimensional vector representations of users to boost the subsequent social network analysis. We assume the dimensions of latent interests of users and latent features of items are the same, which is denoted by  $D$ .

SBRNE is built on two observations as follows: (1) Users with more common hidden interests are more probably having a social association between each other; (2) Users with more common hidden interests are more probably giving similar ratings on items with similar hidden features.

We assume that users' latent interests  $U \in \mathbb{R}^{D \times N}$  and items' latent features  $V \in \mathbb{R}^{D \times M}$  follow two zero-mean spherical Gaussian distributions with hyper-parameters  $\sigma_U^2$  and  $\sigma_V^2$ , respectively. Formally,

$$p(U|\sigma_U^2) = \prod_{i=1}^N \mathcal{N}(U_i|0, \sigma_U^2 I) \quad (6)$$

$$p(V|\sigma_V^2) = \prod_{j=1}^M \mathcal{N}(V_j|0, \sigma_V^2 I) \quad (7)$$

where  $U_i$  and  $V_j$  represent the latent variable for user  $i$  and the latent variable for item  $j$ , respectively.

The social information  $S$  is determined by users' latent interests  $U$  with adding a Gaussian noise with variance  $\sigma_S^2$ , as follows:

$$p(S|U, \sigma_S^2) = \prod_{i=1}^N \prod_{k=1}^N [\mathcal{N}(S_{ik}|g(U_i^T U_k), \sigma_S^2)]^{I_{ik}^S} \quad (8)$$

where  $I^S$  denotes the indication function of social association.  $I_{ik}^S = 1$  if there is an edge between user  $i$  and user  $k$ , and  $I_{ik}^S = 0$  otherwise. In addition, we use the sigmoid function,  $g(x) = \frac{1}{1+e^{-x}}$ , to bound the range of  $U_i^T U_k$  within  $[0, 1]$ .

The behavior information  $R$  between users and items is determined by latent variables  $U$  and  $V$  with adding a Gaussian noise with variance  $\sigma_R^2$ . Formally,

$$p(R|U, V, \sigma_R^2) = \prod_{i=1}^N \prod_{j=1}^M [\mathcal{N}(R_{ij}|g(U_i^T V_j), \sigma_R^2)]^{I_{ij}^R} \quad (9)$$

where  $I^R$  is the indication function of behavior information.  $I_{ij}^R = 1$  if there is a rating made by user  $i$  on item  $j$ , and  $I_{ij}^R = 0$  otherwise. Like modeling the social connections, sigmoid function  $g(x)$  is also utilized to bound the ranges  $U_i^T V_j$  within  $[0, 1]$ . Accordingly, the range of rating scores  $1, \dots, K$  is also transformed into  $[0, 1]$  by using the function  $t(x) = (x - 1)/(K - 1)$ .

### 2.3 Optimization Algorithm

To solve the model conveniently, we use the log-likelihood of the complete-data probability of observed and latent variables as follows.

$$\begin{aligned} \ln p(R, S, U, V | \sigma_U^2, \sigma_V^2, \sigma_S^2, \sigma_R^2) \\ = \ln p(S|U, \sigma_S^2) + \ln p(R|U, V, \sigma_R^2) + \ln p(U|\sigma_U^2) + \ln p(V|\sigma_V^2) \end{aligned} \tag{10}$$

An alternative updating procedure is utilized to compute the model: (i) a point estimation of latent variables  $U$  and  $V$  is updated with keeping hyper-parameters fixed; (ii) updates of hyper-parameters are derived with keeping latent variables  $U$  and  $V$  as constants. The two steps are alternated until a termination condition is met.

For brevity, we use  $LL$  as a surrogate of Eq. (10) in the following analysis.

**Point-Estimation of Latent Variables.** To derive the updates of latent variables, we calculate the partial derivatives of  $LL$  w.r.t.  $U$  and  $V$  with keeping hyper-parameters fixed.

For each  $i \in \{1, \dots, N\}$ , the partial derivative of  $U_i$  is derived as follows.

$$\begin{aligned} \frac{\partial LL}{\partial U_i} &= \frac{1}{\sigma_S^2} \sum_{k=1}^N I_{ik}^S (S_{ik} - g(U_i^T U_k)) g'(U_i^T U_k) U_k \\ &+ \frac{1}{\sigma_R^2} \sum_{j=1}^M I_{ij}^R (R_{ij} - g(U_i^T V_j)) g'(U_i^T V_j) V_j - \frac{1}{\sigma_U^2} U_i \end{aligned} \tag{11}$$

where  $g'(\cdot)$  is the derivative of sigmoid function  $g(\cdot)$ , and the same hereinafter.

Likewise, the partial derivative of  $V_j$  for  $j \in \{1, \dots, M\}$  is derived as follows.

$$\frac{\partial LL}{\partial V_j} = \frac{1}{\sigma_R^2} \sum_{i=1}^N I_{ij}^R (R_{ij} - g(U_i^T V_j)) g'(U_i^T V_j) U_i - \frac{1}{\sigma_V^2} V_j \tag{12}$$

Since the coupling of latent variables for different users and items in Eqs. (11) and (12), we know that we can't obtain a closed form of updates of  $U$  and  $V$ . The iterative updates of the latent variables are used as follows.

$$U_i^{new} = U_i^{old} + \rho_1 \frac{\partial LL}{\partial U_i} \tag{13}$$

$$V_j^{new} = V_j^{old} + \rho_2 \frac{\partial LL}{\partial V_j} \tag{14}$$

where  $U^{old}$  and  $V^{old}$  are latent variables derived in the last iteration, and  $\rho_1$  and  $\rho_2$  are the learning rates for latent variable  $U$  and  $V$ , respectively.

**Updates of Hyper-Parameters.** To obtain the updates of hyper-parameters, we calculate the partial derivatives of  $LL$  *w.r.t.* hyper-parameters with treating  $U$  and  $V$  as constants. It's worth noting that the updates of hyper-parameters can be obtained as a closed-form by making the corresponding partial derivatives as zero. Specifically, the updates of hyper-parameters are derived as follows.

$$\sigma_S^2 = \frac{\sum_{i=1}^N \sum_{k=1}^N I_{ik}^S (S_{ik} - g(U_i^T U_k))^2}{\sum_{i=1}^N \sum_{k=1}^N I_{ik}^S} \quad (15)$$

$$\sigma_R^2 = \frac{\sum_{i=1}^N \sum_{j=1}^M I_{ij}^R (R_{ij} - g(U_i^T V_j))^2}{\sum_{i=1}^N \sum_{j=1}^M I_{ij}^R} \quad (16)$$

$$\sigma_U^2 = \frac{\sum_{i=1}^N U_i^T U_i}{DN} \quad (17)$$

$$\sigma_V^2 = \frac{\sum_{j=1}^M V_j^T V_j}{DM} \quad (18)$$

## 2.4 Time Complexity Analysis

To analyze the time complexity of SBRNE, we denote numbers of non-zero entries of  $S$  and  $R$  as  $n$  and  $m$ , respectively. Denote also the dimension of users' interests and item features as  $D$ , the number of iterations as  $t$  in network embedding procedure, and the number of iterations as  $l$  in optimization procedure. The whole algorithm consists of two parts: network embedding procedure and optimization procedure.

**Network Embedding.** In network embedding, the structural similarity of  $n$  edges in the social network is calculated via Eq. (1), with the time complexity  $O(nd_{ave})$  where  $d_{ave}$  is the average degree of nodes in the social network. For a real-world network, the node degree distribution follows the power-law distribution, which means the average degree of nodes is much less than the number of nodes.

In each iteration of updating the low-dimensional representations of users, the main computation is to calculate the gradient of the objective *w.r.t.*  $U$  via Eq. (4). For each edge, the two adjacent nodes will both occur once in summation of right-hand side of Eq. (4). Therefore, the total number of items in summation of right-hand side of Eq. (4) is  $2n$ , and for each item, the time complexity of calculation is  $O(D)$  (for the multiplication of a scalar by a  $D$ -dimensional vector). For the whole network, the time complexity of each iteration is  $O(Dn)$ . Therefore, the time complexity of network embedding is  $O(tDn + d_{ave}n)$ , which is linear to the number of edges in social network. We usually set  $D$  as the same magnitude with  $d_{ave}$ , and the time complexity can be denoted compactly as  $O(tDn)$ .

For the optimization procedure, the main operations in each iteration include updates of latent variables and hyper-parameters via Eqs. (13)–(18).



**Optimization of Latent Variables.** To compute derivatives of latent variables  $U$  by Eq. (11), the main operations consist of  $n$  times of inner product of two latent variables  $U_i$  and  $U_k$  (for each non-zero entry of  $S$ ), and  $m$  times of inner product of two latent variables  $U_i$  and  $V_j$  (for each non-zero entry of  $R$ ). Therefore, the time complexity in update of  $U$  is  $O((n+m)D)$ .

To compute derivatives of latent variables  $V$  by Eq. (12), the main operations consist of  $m$  times of inner product of two latent variables  $U_i$  and  $V_j$  (for each non-zero entry of  $R$ ) with the time complexity in update of  $V$  is  $O(mD)$ .

**Optimization of Hyper-Parameters.** In Eq. (15), the denominator is  $n$  trivially. Hence, it takes  $n$  times of inner product of two latent variables  $U_i$  and  $U_k$  (for numerator of right-hand side in Eq. (15)) with time complexity  $O(nD)$ , to update of hyper-parameter  $\sigma_S^2$ . Likewise, the time complexity of update of hyper-parameter  $\sigma_R^2$  is  $O(mD)$ .

To update hyper-parameter  $\sigma_U^2$ , it takes  $N$  times of inner product of  $U_i$  ( $i \in \{1, \dots, N\}$ ) and itself with time complexity  $O(ND)$ . Likewise, the time complexity of update of hyper-parameter  $\sigma_V^2$  is  $O(MD)$ .

Based on above analyses for network embedding, latent variables and hyper-parameters, the time complexity of our algorithm is  $O(tDn+lD(n+m+N+M))$ . Usually,  $n > N$  and  $m > M$ , which means in average each user has at least one social connection and each item has at least one rating. The time complexity can be expressed compactly as  $O(tDn+lD(n+m))$ , *i.e.*, it's linear in the summation of numbers of non-zero entries in  $S$  and  $R$ .

## 3 Experiments

### 3.1 Experimental Setup

Our experiments consist of two parts. Firstly, we evaluate SBRNE on social and behavior recommendation tasks on three real-world datasets, Epinions<sup>1</sup>, Ciao<sup>2</sup>, and DBLP<sup>3</sup>. Secondly, we evaluate the recommendation performance of SBRNE on cold-start users, *i.e.* users with very few social links or ratings.

**Real-World Datasets and Preprocessing.** Originally two datasets, Epinions and Ciao, consist of social relationships among users and users' rating on products with scores from 1 to 5. In the preprocessing procedure, we remove items with less than 3 ratings. Social connections between pairs of users are treated as undirected unweighted edges, *i.e.* directions of original social connections are ignored, and if two users have a connection, the social weight is 1; otherwise, the social weight is 0. Finally, for Epinions, there are 285,788 social edges among

<sup>1</sup> <http://www.epinions.com>.

<sup>2</sup> <http://www.ciao.co.uk>.

<sup>3</sup> <https://dblp.uni-trier.de>.

18,062 users, and 525,053 ratings on 58,995 items. For Ciao, there are 85,011 social edges among 7,282 users, and 183,332 ratings on 21,881 items.

For DBLP, authors are viewed as users and extracted meaningful words (by a common NLP preprocessing procedure) in titles are treated as items. The original social network (*i.e.* co-author network) and behavior information (bipartite network between authors and words) are all weighted networks, having weights representing the number of co-authored papers between author pairs and the times of the word used by the author in titles of her/his authorized papers. Authors with less than 20 publications and words with less than 50 occurrences are removed from original dataset. To bound the social and behavior information in the same range as that of other two datasets, we utilize a normalization procedure and a discretization procedure on social and behavior information, respectively.

We normalize the social information by cosine similarity. Assume the weighted co-author network be denoted by a matrix  $W \in \mathbb{R}^{N \times N}$ , where  $N$  denotes the number of users, and  $W_{ik}$  is the number of co-authored papers by authors  $i$  and  $k$ . For each  $i \in \{1, \dots, N\}$ ,  $W_{ii}$  is defined as 0. For each pair of authors  $i$  and  $k$ , the final social weight is normalized as follows.

$$S_{ik} = \frac{W_{ik}}{\sqrt{\sum_{t=1}^N W_{it} \cdot \sum_{t=1}^N W_{kt}}} \quad (19)$$

For each pair of an author and a word, the number of occurrences of the word used by the author is mapped into  $\{1, 2, 3, 4, 5\}$  by a discretization procedure. Based on our preliminary results, intervals of occurrences  $[1, 5]$ ,  $(5, 10]$ ,  $(10, 20]$ ,  $(20, 30]$ , and  $(30, +\infty)$  are mapped into 1, 2, 3, 4, and 5, respectively. Finally, the statistics of three datasets are listed in Table 1.

**Table 1.** Real-world datasets

Dataset	Epinions	Ciao	DBLP
#Users	18,062	7,282	98,692
#Items	58,995	21,881	11,678
#Ratings	525,053	183,332	16,422,690
Rating density	0.0005	0.0012	0.0142
#Social	285,788	85,011	1,407,256
Social density	0.0018	0.0032	0.0003

For each dataset, we choose randomly 80% of it (both social and behavior information) as a training set, leaving the remainder as the hold-out testing set. We split further the training data into four equal-sized subsets for 4-fold cross validation. The parameters deriving the best performance on the hold-out testing set are used for performance evaluation. Finally, the dimension of latent variables

$D = 10$ , learning rate  $\rho = 10^{-4}$  and number of iterations  $t = 20$  (for network embedding), learning rates  $\rho_1 = \rho_2 = 10^{-4}$  and number of iterations  $l = 30$  (for optimization procedure). For each experiment, the 4-fold cross-validation is repeated 20 times, with each time the order of the instances in each dataset being randomized. The average measurement and standard deviation are calculated for recommendation performance evaluation.

**Baselines.** Due to few previous work (as far as we know, only GFBM [1]) addresses user and item recommendations simultaneously, we compare SBRNE with two groups of baselines in user recommendation and item recommendation respectively to show the performance improvement. To evaluate the effectiveness of network embedding, we also implement a model (denoted by SBR\_Rand) which replaces the network embedding procedure with a random initialization of latent variables  $U$  in SBRNE.

For user recommendation, baselines are listed as follows:

- *Cosine* and *Jaccard* only take into account the common neighbors to evaluate the probability of having a social connections between user pairs.
- *COP* [24] considers both social and behavior information to evaluate the co-occurrence probability of the corresponding pair of users.
- *PropFlow* [11] considers paths between user pairs to evaluates the probability of the potential links between them.
- *GFBM* [1] is a generative framework for both user and item recommendations based on a Bayesian model.

For item recommendation, selected baselines include<sup>4</sup>:

- *PMF* [21] is the seminal probabilistic latent factor model for item recommendation, which only considers the behavior information.
- *SoRec* [18] employs both social network and rating records, which models users’ profiles with two latent random variables.
- *LOCABAL* [22] exploits local and global social information for item recommendation.
- *GFBM* [1] is a generative framework for both user and item recommendations based on a Bayesian model.
- *PRMF* [14] learns the dependencies between users to improve the performance on item recommendation.
- *PTPMF* [26] exploits the strong and weak ties between users to improve the performance on item recommendation.

We use two commonly used metrics, Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) [22,26], to evaluate the recommendation performance of SBRNE and the selected baselines. A smaller RMSE or MAE value indicates a better recommendation performance.

---

<sup>4</sup> We didn’t compare SBRNE with SREPS [13], since we failed to find an implementation of it.

### 3.2 Evaluation of Social Recommendation

Table 2 provides the following observations:

- Cosine and Jaccard exhibit similar and the worst performance on user recommendation among all of the three datasets. It makes sense that both two methods exploit only the one-step topological structure of network (*i.e.* neighbors of users) for user recommendation.
- PropFlow derives better results than those of Cosine and Jaccard. The reason is that PropFlow defines similarity between pairs of users by considering multiple-steps topological structure of network (*i.e.* paths between users), which may provide more information than neighbors of users.
- COP, GFBM, SBR\_Rand, and SBRNE which all consider both social and behavior information, performs better than other methods on all of three datasets, indicating the importance of users’ behavior information for improving the performance of user recommendation. Moreover, COP, GFBM and SBR\_Rand derive comparable performance, which is worse than that of

**Table 2.** Comparisons on social recommendation (boldface font denotes the best performance)

Datasets	Method	RMSE	MAE
Epinions	Cosine	0.2985 ± 0.0022	0.3119 ± 0.0021
	Jaccard	0.3001 ± 0.0019	0.3210 ± 0.0022
	COP	0.1804 ± 0.0023	0.2237 ± 0.0021
	PropFlow	0.2011 ± 0.0019	0.2416 ± 0.0018
	GFBM	0.1787 ± 0.0023	0.2207 ± 0.0023
	SBR_Rand	0.1803 ± 0.0027	0.2327 ± 0.0023
	SBRNE	<b>0.1716</b> ± 0.0021	<b>0.2113</b> ± 0.0021
Ciao	Cosine	0.2815 ± 0.0019	0.3011 ± 0.0018
	Jaccard	0.2922 ± 0.0022	0.3087 ± 0.0023
	COP	0.1623 ± 0.0022	0.1977 ± 0.0022
	PropFlow	0.1802 ± 0.0023	0.2103 ± 0.0022
	GFBM	0.1603 ± 0.0023	0.1949 ± 0.0021
	SBR_Rand	0.1737 ± 0.0024	0.1975 ± 0.0023
	SBRNE	<b>0.1577</b> ± 0.0021	<b>0.1800</b> ± 0.0021
DBLP	Cosine	0.3122 ± 0.0022	0.3600 ± 0.0018
	Jaccard	0.3345 ± 0.0021	0.3619 ± 0.0022
	COP	0.2225 ± 0.0021	0.2466 ± 0.0021
	PropFlow	0.2389 ± 0.0018	0.2601 ± 0.0024
	GFBM	0.2373 ± 0.0022	0.2473 ± 0.0024
	SBR_Rand	0.2253 ± 0.0022	0.2478 ± 0.0024
	SBRNE	<b>0.2213</b> ± 0.0019	<b>0.2459</b> ± 0.0022

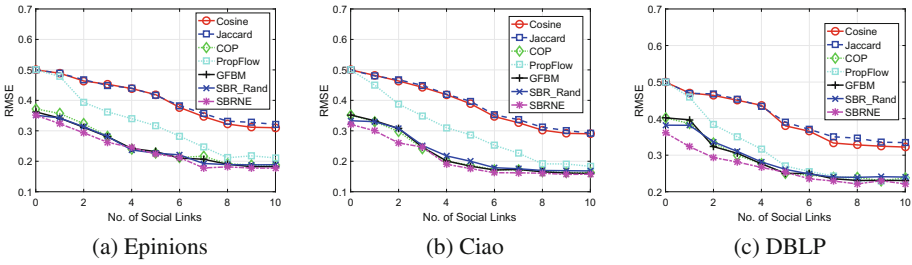


Fig. 1. Comparisons of social recommendation on cold-start users

SBRNE. The reason is that network embedding is able to make full use of social information to derive a better users’ latent profiles which can improve the user recommendation performance.

- The performance difference on DBLP between of SBR\_Rand and SBRNE is smaller than those on other two datasets. One possible reason is that the normalization procedure of DBLP by cosine similarity plays a similar role with network embedding in SBRNE. The other reason lies in that the extremely sparse social connections (with social density about 0.0003 on DBLP) may affect the effectiveness of network embedding.

### 3.3 Evaluation of Behavior Recommendation

The comparison results of behavior recommendation are presented in Table 3. It’s worth to point out that, due to the randomness in data splitting and model initialization as well as possible differences in preprocessing procedures, our results derived from some baselines are slightly different from the results reported in original papers.

- PMF performs worse than all of the remainders which utilize both social and behavior information for item recommendation. This observation verifies our assumption that social information indeed helps for improving performance on item recommendation.

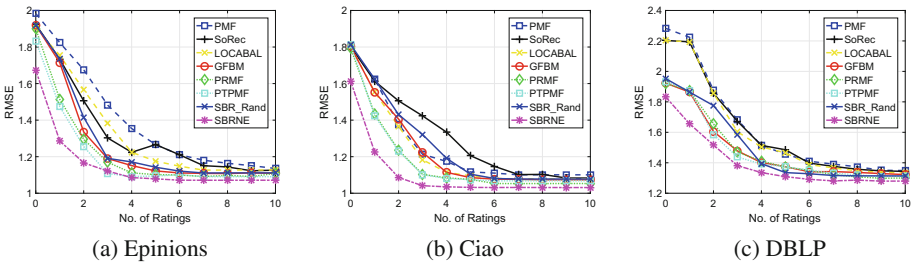


Fig. 2. Comparisons of behavior recommendation on cold-start users

- Through drilling down further the results, we observe that SoRec, LOCAL, GFBM, PRMF and SBR\_Rand derive comparable results on all of three datasets, and all of them perform a little bit worse than PTPMF. However, Table 3 shows also that PTPMF exhibits relatively higher standard deviation on performance over three datasets, which means PTPMF performs not stable on item recommendation. The possible reason is that it’s not trivial to learn a reasonable threshold for classifying strong and weak ties in PTPMF, and an inappropriate threshold might lead to a sub-optimal item recommendation result.
- SBRNE performs best on almost all of three datasets. One reason is that it models social and behavior information in a unified framework, which makes full use of all available information in a reasonable way. Actually, from the objective in Eq. (10), we can find that social recommendation related term  $\ln p(S|U, \sigma_S^2)$  can be viewed as a regularization term for behavior recommendation related terms (the remaining parts in Eq. (10)), *i.e.* results from social recommendation can help to improve the performance on item recommendation. Likewise, results from item recommendation can also help to improve the performance on social recommendation. The other reason is that the network embedding helps to learn a more meaningful latent users’ profiles which can improve the item recommendation performance.

### 3.4 Evaluation on Cold-Start Users

In this section, we investigate the recommendation performance of SBRNE on cold-start users, *i.e.* users with relatively few social links or ratings. More specifically, we compare social and behavior recommendation performance of SBRNE with selected baselines on users with  $\leq 10$  social links/ratings. We evaluate the average recommendation performance of all methods on users with different number of social links/ratings, and the results are presented in Figs. 1 and 2. We want to note that comparison results exhibit similar patterns according to both RMSE and MAE, and we only presents the comparison result on RMSE due to the limit of space.

Results demonstrate that SBRNE outperforms the baselines on both social recommendation and behavior recommendation for cold-start users. More specifically, SBRNE is able to derive an acceptable recommendation performance on users with extremely few ( $\leq 5$ ) social links/ratings. We believe that the stable performance is mainly due to the integration of social information and behavior information into a unified framework. Comparing with SBR\_Rand show that the proposed network embedding method can improve the recommendation performance on cold-start users. In addition, the recommendation performance of SBRNE becomes stable on users with about more than 5 social links/ratings.

**Table 3.** Comparisons of Behavior Recommendation (boldface font denotes the best performance)

Datasets	Method	RMSE	MAE
Epinions	PMF	1.1352 ± 0.0033	0.9513 ± 0.0021
	SoRec	1.1273 ± 0.0027	0.9223 ± 0.0022
	LOCABAL	1.1279 ± 0.0025	0.9222 ± 0.0019
	GFBM	1.1253 ± 0.0031	0.9205 ± 0.0025
	PRMF	1.1097 ± 0.0032	0.8913 ± 0.0022
	PTPMF	1.0907 ± 0.0038	0.8812 ± 0.0027
	SBR_Rand	1.1175 ± 0.0025	0.9057 ± 0.0021
	SBRNE	<b>1.0713</b> ± 0.0025	<b>0.8593</b> ± 0.0017
Ciao	PMF	1.1009 ± 0.0026	0.8836 ± 0.0022
	SoRec	1.0789 ± 0.0027	0.8202 ± 0.0023
	LOCABAL	1.0784 ± 0.0029	0.8165 ± 0.0019
	GFBM	1.0777 ± 0.0031	0.8163 ± 0.0021
	PRMF	1.0631 ± 0.0028	0.7984 ± 0.0023
	PTPMF	1.0577 ± 0.0030	0.7982 ± 0.0025
	SBR_Rand	1.0712 ± 0.0025	0.8016 ± 0.0020
	SBRNE	<b>1.0312</b> ± 0.0026	<b>0.7913</b> ± 0.0019
DBLP	PMF	1.3437 ± 0.0032	1.2313 ± 0.0028
	SoRec	1.3337 ± 0.0033	1.2174 ± 0.0028
	LOCABAL	1.3211 ± 0.0032	1.2122 ± 0.0025
	GFBM	1.3137 ± 0.0031	1.2124 ± 0.0029
	PRMF	1.2988 ± 0.0033	1.0912 ± 0.0031
	PTPMF	<b>1.2861</b> ± 0.0035	<b>1.0369</b> ± 0.0033
	SBR_Rand	1.3141 ± 0.0032	1.2172 ± 0.0025
	SBRNE	<b>1.2861</b> ± 0.0031	1.0377 ± 0.0023

## 4 Related Work

Social recommendation, *a.k.a.* link prediction, is to compute a similarity between each pair of users to derive a ranking list of links or users for recommendation. Some methods only take into account one-step social information of network, including Cosine similarity [27] and Jaccard similarity [5]. A few methods consider multiple-steps social information or paths between nodes, among which PropFlow [11] is a representative one. Some methods consider both social and behavior information for link prediction, *e.g.* Co-Occurrence Probability (COP) [24].

In recent years, quite a lot of studies investigate the item recommendation problem [28]. The Probabilistic Matrix Factorization (PMF) [21] is the first probabilistic latent factor model to model users' ratings on items, which only

considers the behavior information for item recommendation. After the seminal work, many different extensions of PMF have been proposed, considering not only the behavior information but also the social information, including SoRec model [18], Social Regularization model [19], LOCABAL [22], PRMF [14], PTPMF [26], SREPS [13], etc. Most of the extended models utilize more than one random variables to describe users' latent profiles, leading to a more complex model which may affect the performance of item recommendation.

GFBM [1] is the most related work to our proposed model, which addresses both user and item recommendations under a unified model. GFBM models each user with two notions, susceptibility and expertise, advancing the expressiveness of users at the cost of introducing more noises which may lead to undesirable recommendation results.

Compared with previous studies, SBRNE is able to address both user and item recommendations under a unified, simple but effective model, which takes network embedding as a pre-training procedure for users' representations. With SBRNE, performance of social recommendation and behavior recommendation can be improved mutually, which has been demonstrated in the experimental section.

## 5 Conclusions and Future Work

In this paper, we have proposed a unified framework SBRNE which models social and behavior information simultaneously and utilizes network embedding as a pre-training step for users' profiles, leading to a performance improvement both on social and behavior recommendation tasks. In SBRNE, social information is determined by users' latent profiles with adding a Gaussian noise, while behavior information is determined by users' latent profiles and items' latent features with adding also a Gaussian noise. Through using users' latent profiles as a bridge, both tasks of social and behavior recommendation can be improved mutually under the unified framework. In addition, we have devised a network embedding procedure to pre-train users' profiles to advance the model fitness. The comprehensive experiments on real-world datasets show that: (1) SBRNE outperforms the selected baselines on both social and behavior recommendation tasks; (2) SBRNE performs stable on recommendation tasks for cold-start users; (3) The network embedding procedure can improve the recommendation performance of the proposed framework.

There are two directions to investigate in the future. First, we are interested in extending the framework to treat the evolving of social network, *i.e.* the social network may change itself dynamically. Second, we would like to extend the framework to a full Bayesian model, which estimates a distribution for each hyper-parameter and might derive a more stable performance.

**Acknowledgments.** The work is partially supported by the National Natural Science Foundation of China (Nos. 61802404, 61762078, 61702508, 61663004, 61602438), the CCF-Tencent Rhino-Bird Young Faculty Open Research Fund (No. RAGR20180111). Authors are grateful to the anonymous reviewers for helpful comments.






## References

1. Costa, G., Manco, G., Ortale, R.: A generative Bayesian model for item and user recommendation in social rating networks with trust relationships. In: Calders, T., Esposito, F., Hüllermeier, E., Meo, R. (eds.) ECML PKDD 2014, Part I. LNCS (LNAI), vol. 8724, pp. 258–273. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-44848-9\\_17](https://doi.org/10.1007/978-3-662-44848-9_17)
2. Cui, P., Wang, X., Pei, J., Zhu, W.: A survey on network embedding. *IEEE Trans. Knowl. Data Eng.* (2018). <https://doi.org/10.1109/TKDE.2018.2849727>
3. Getoor, L., Diehl, C.: Link mining: a survey. *ACM SIGKDD Explor. Newsl.* **7**(2), 3–12 (2005)
4. Guo, G., Zhang, J., Yorke-Smith, N.: TrustSVD: collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In: Proceedings of AAAI, pp. 123–129 (2015)
5. Hasan, M.A., Zaki, M.J.: A survey of link prediction in social networks. In: Aggarwal, C. (ed.) *Social Network Data Analytics*, pp. 243–275. Springer, Boston (2011). [https://doi.org/10.1007/978-1-4419-8462-3\\_9](https://doi.org/10.1007/978-1-4419-8462-3_9)
6. Hu, G., Dai, X., Huang, Y.S.S., Chen, J.: A synthetic approach for recommendation: combining ratings, social relations, and reviews. In: Proceedings of IJCAI, pp. 1756–1762 (2015)
7. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: Proceedings of ICDM, pp. 263–272 (2008)
8. Jamali, M., Ester, M.: A matrix factorization technique with trust propagation for recommendation in social networks. In: Proceedings of RecSys, pp. 135–142 (2010)
9. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. *Computer* **42**(8), 30–37 (2009)
10. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. *J. Am. Soc. Inf. Sci. Technol.* **58**(7), 1019–1031 (2007)
11. Lichtenwalter, R., Lussier, J., Chawla, N.: New perspectives and methods in link prediction. In: Proceedings of KDD, pp. 243–252 (2010)
12. Linden, G., Smith, B., York, J.: Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Comput.* **7**(1), 76–80 (2003)
13. Liu, C., Zhou, C., Wu, J., Hu, Y., Guo, L.: Social recommendation with an essential preference space. In: Proceedings of AAAI, pp. 346–353 (2018)
14. Liu, Y., Zhao, P., Liu, X., Wu, M., Duan, L., Li, X.: Learning user dependencies for recommendation. In: Proceedings of IJCAI, pp. 2379–2385 (2017)
15. Liu, Y., Zhao, P., Sun, A., Miao, C.: A boosting algorithm for item recommendation with implicit feedback. In: Proceedings of IJCAI, pp. 1792–1798 (2015)
16. Ma, H.: An experimental study on implicit social recommendation. In: Proceedings of SIGIR, pp. 73–82 (2013)
17. Ma, H.: On measuring social friend interest similarities in recommender systems. In: Proceedings of SIGIR, pp. 465–474 (2014)
18. Ma, H., Yang, H., Lyu, M., King, I.: SoRec: social recommendation using probabilistic matrix factorization. In: Proceedings of CIKM, pp. 931–940 (2008)
19. Ma, H., Zhou, D., Liu, C., Lyu, M., King, I.: Recommender systems with social regularization. In: Proceedings of WSDM, pp. 287–296 (2011)
20. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. In: Proceedings of ACM SIGKDD, pp. 701–710, August 2014
21. Salakhutdinov, R., Mnih, A.: Probabilistic matrix factorization. In: Proceedings of NIPS, pp. 1257–1264 (2007)

22. Tang, J., Hu, X., Gao, H., Liu, H.: Exploiting local and global social context for recommendation. In: Proceedings of IJCAI, pp. 2712–2718 (2013)
23. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: Proceedings of WWW, pp. 1067–1077, May 2015
24. Wang, C., Satuluri, V., Parthasarathy, S.: Local probabilistic models for link prediction. In: Proceedings of ICDM, pp. 322–331 (2007)
25. Wang, X., Donaldson, R., Nell, C., Gorniak, P., Ester, M., Bu, J.: Recommending groups to users using user-group engagement and time-dependent matrix factorization. In: Proceedings of AAAI, pp. 1331–1337 (2016)
26. Wang, X., Hoi, S., Ester, M., Bu, J., Chen, C.: Learning personalized preference of strong and weak ties for social recommendation. In: Proceedings of WWW, pp. 1601–1610 (2017)
27. Xu, X., Yuruk, N., Feng, Z., Schweiger, T.: Scan: a structural clustering algorithm for networks. In: Proceedings of KDD, pp. 824–833 (2007)
28. Yang, X., Guo, Y., Liu, Y., Steck, H.: A survey of collaborative filtering based social recommender systems. *Comput. Commun.* **41**, 1–10 (2014)
29. Yu, Z., Wang, C., Bu, J., Wang, X., Wu, Y., Chen, C.: Friend recommendation with content spread enhancement in social networks. *Inf. Sci.* **309**, 102–118 (2015)
30. Zhao, T., McAuley, J., King, I.: Leveraging social connections to improve personalized ranking for collaborative filtering. In: Proceedings of CIKM, pp. 261–270 (2014)



# User Intention-Based Document Summarization on Heterogeneous Sentence Networks

Hsiu-Yi Wang<sup>1</sup>, Jia-Wei Chang<sup>2</sup> , and Jen-Wei Huang<sup>3</sup>  

<sup>1</sup> Institute of Computer and Communication Engineering,  
National Cheng Kung University, Tainan City 701, Taiwan  
wendywang0621@gmail.com

<sup>2</sup> Department of Computer Science and Information Engineering,  
National Taichung University of Science and Technology, Taichung City 404, Taiwan  
jiaweichang.gary@gmail.com

<sup>3</sup> Department of Electrical Engineering, National Cheng Kung University,  
Tainan City 701, Taiwan  
jwhuang@mail.ncku.edu.tw

**Abstract.** Automatic extraction-based document summarization is a difficult Natural Language Processing task. Previous approaches have usually generated the summary by extracting the top  $K$  salient sentences on graph-based ranking algorithms, but sentence feature representation only captures the surface relationship between the objects, hence the results may not accurately reflect the user's intentions. Therefore, we propose a method to address this challenge, and: (1) obtain deeper semantic concepts among candidate sentences using meaningful sentence vectors combining word vectors and TF-IDF; (2) rank the sentences considering both relationships between sentences and the user's intention for each sentence to identify significant sentences, and apply these to a heterogeneous graph; (3) generate the result sentence by sentence to ensure summary semantics are properly related to the original document. We verified the proposed approach experimentally using English summarization benchmark datasets DUC2001 and DUC2002; the large Chinese summarization data set, LCSTS. We also collected news data and produced a reference summary using a group of bank auditor experts that we compared to the proposed approach using ROUGE evaluation.

**Keywords:** Document summarization ·  
Heterogeneous sentence networks ·  
TF-IDF based deep word representation

## 1 Introduction

Text summarization in opinion mining, also called opinion summarization, can assist people in decision making [11]. Opinion summarization can capture the opinion-oriented information from people's opinions on various subjects in the

massive amounts of textual data [6], and then which helps people reduce efforts and make the work more efficient. Therefore, text summarization considering the user intention by user’s keyword can provide more useful and valuable applications. In essence, opinion summarization is one of the summary types, thus which is very depending on the text summarization methods [1]. The main task of text summarization methods is to determine the important information and eliminate the redundant information.

Many unsupervised summarization methods used the concept of graph-based ranking, such as PageRank [15]. For example, LexRank [4], TextRank [13] and DivRank [12] depend on the ranking concept and consider the prestige and semantic diversity of sentences. However, these methods generate an objective summary rather than a summary based on user intention. That is, the summary may not include the interesting sentences by users if the users’ keywords are not important to the original texts. For instance, the bank auditor needs a credit risk summary of a certain company from the news. The summaries of these methods may not include the sentences that related to the credit risk events, even though their summaries can represent the objective description of the news.

Therefore, we aim to develop a keyword-based summarization algorithm using the heterogeneous networks which include the sentences from a document and the given keywords from the user. The proposed algorithm can extract the textual units which have highly related to the given keywords and automatically generate a summary based on the user intention.

## 2 Graph-Based Summarization Methods

The graph-based summarization methods have two phases [7], textual unit ranking and textual unit selection. For textual unit selection, how to choose the salient and non-redundant unit to generate a salience summary. Graph-based ranking models are popularly used in summarization task. Many related approaches are inspired by PageRank. For example, LexRank [4] is a famous stochastic graph-based approach, which uses PageRank to measure the importance of sentence and extracts the most important ones to include in the summarization. LexRank also adopts the cross-sentence information subsumption as the redundancy removal method. Thus, it can both balance the low redundancy and high prestige of sentences to avoid selecting too many similar sentences into the summary. In addition, DivRank [12] using a time-variant random walk aims to balance the prestige and diversity of sentences in ranking. For improving the semantic representation, DivSelect+CNNLM [16] developed an unsupervised convolutional neural network (CNN) to learn sentence representations, and which proposed a new sentence selection algorithm to balance sentence prestige and diversity.

To extract sentence which related to user intention, we can add different intention unit into the original homogeneous graph to optimize important sentences. However, the graph-based ranking methods mention above cannot solve the problem when we have various type objects and relations in the heterogeneous graph. Therefore, we propose *HeteroRank* to produce Top  $K$  important sentences with diversified intention unit and relationship.

### 3 Methods of Semantic Representation

Word representation also be called word embedding, is a type of mapping that use vectors to represent words and allows to capture semantic relationship between the words. In 2013, the more popular of the word embedding model CBOW (Continuous Bag-of-Words Model) and Skip-gram was proposed by Mikolov, Tomas [14]. The method will train the model by computing the continuous distributed representation for the word. Besides, after training of the model, we can get the word vectors as word representation and obtain significant information by project word vectors into vector space.

We can understand how to train and learn the word vector by Eq. 1. Word2vec is a single hidden layer and fully connected neural network. One-hot vectors of context words  $w_{context} = w^1, w^2, \dots, w^n$  will be the input of training words and the objective function is to maximize probabilities of target words  $w_{target}$  in the output layer.

The equation is as below:

$$\min_{W, W'} - \sum_{t=1}^{n-1} \log P(w_{t+1} | w_t) \quad (1)$$

where:

$$Pr(w_{target} | w_{context}) = \frac{e^{y_{w_{target}}}}{\sum_{i=1}^n e^{y_i}} \quad (2)$$

$$y = W'^T h, h = W^T x \quad (3)$$

For CBOW, model predicting the upcoming target word in current context words. For instance, we can use “monkey” and “tree” as context words for “climb” as the target word. Given  $w_{i-1}, w_{i-2}, w_{i+1}, w_{i+2}$  as input, the output of model will be word  $w_i$ . However, Skip-gram model reverses the use of target and context words. In this case, the target word is fed at the input to predict the context words. Taking the example of “monkey” and “tree” as target words and “climb” as the context word. Given  $w_i$  as input, the output of model will be word  $w_{i-1}, w_{i-2}, w_{i+1}, w_{i+2}$ .

## 4 Methodology

### 4.1 Feature Extraction

We want to get the important sentences in a document where the sentence can also seem as a combination of a series of words. Therefore, based on the concept of word2vec model, our propose method uses pre-train CBOW model to obtain word vector in the document. Each word in the document will be mapped into a unique vector. Then average the word vectors in the sentence can represent as the sentence vector.

Beside the assumption above, in order to capture the important sentences, the model not only use average word vectors to represent sentence vectors but

also include the TF-IDF feature. We can recognize the important words by TF-IDF weight with some limitation and multiply to word's representation which show in Eq. 4. In this way, it will be accurately for us to get the significant sentence.

$$Emb_s(S) = \frac{\sum_{w_i \in S} tfidf(w_i, S, D) * Emb_w(w_i)}{\sum_{w_i \in S} tfidf(w_i, S, D)} \quad (4)$$

where

$$tfidf(w, S, D) = tf_{w,S} * idf_{w,D} \quad (5)$$

$$tf_{w,S} = \frac{n_{w,S}}{\sum_i n_{i,S}} \quad (6)$$

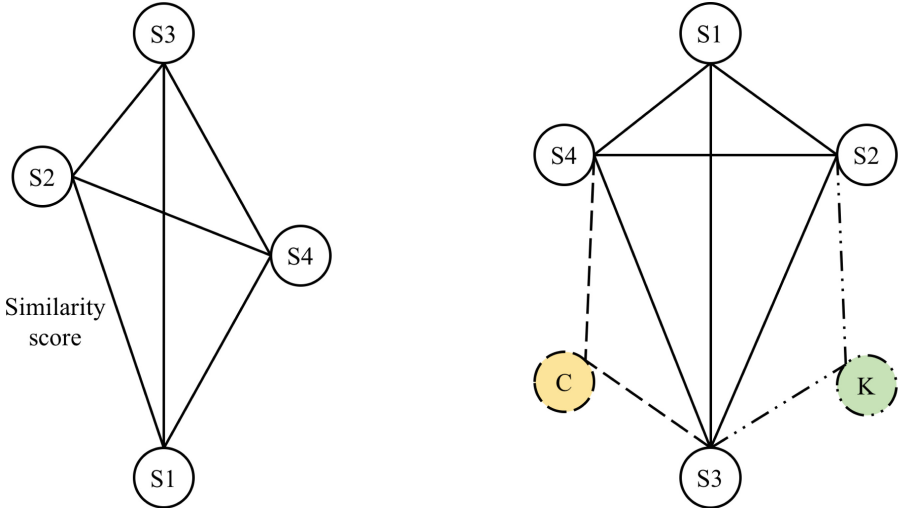
$$idf_{w,D} = \log \frac{|D|}{|\{S \in D : w \in S\}|} \quad (7)$$

## 4.2 HeteroRank

After feature extraction, we apply the representation of sentence on heterogeneous graph. Unlike the previous work which build in original homogeneous graph in Fig. 1(a). In our task, we want to know the relation between original document and user intention like company name and negative keyword. In other words, our goal is to weighted the sentences prestige which have these two textual terms we concentrate. For instance, the sentence consist of company name and negative keyword may have higher priority be chosen in summary for the bank auditor who want to know the company reputation. Therefore, these two textual terms will be added into graph and the graph turns into heterogeneous graph which with different type nodes and edges. In Fig. 1(b), node with  $S_i$  is sentence; node with  $C$  means company name and node with  $K$  represent negative keyword. If company name or negative keyword occur in sentence  $S_i$ , add an edge to represent the relation between them. For instance, dashed edge is company word  $C$  appear in sentence  $S3$  and  $S4$ ; dash-dotted edge means negative word  $K$  write in sentence  $S2$  and  $S3$  in Fig. 1(b).

Therefore, rather than still use original TextRank to deal heterogeneous graph, first thing we need to do is to divide meta-graph and generate meta-path. The concept of meta-path is to explore the semantic and accurately capture the relationships between multiple types of objects in heterogeneous graph. So meta-path with different type object will have distinct semantic meaning. Moreover, the pattern of meta-path is based on user's intention. Under this framework, we can get the best meta-path with pre-define guidance in structure.

Heterogeneous graph in Fig. 1(b) which build by our task can be divided into three meta-graphs. For instance, nodes in the first meta-graph all stand for sentence. However, in second and third meta-graphs, type of the nodes are not the same. The former have sentence and negative keyword node while the latter



(a) edges all present the similarity between sentences

(b) edges represent multiple relations between multiple nodes

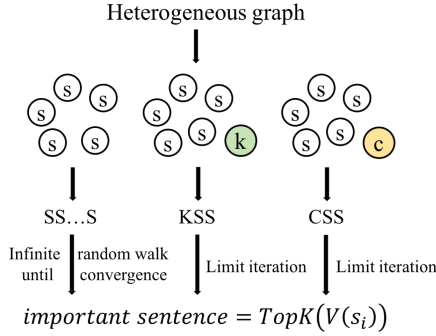
**Fig. 1.** Left is homogeneous graph and right one is heterogeneous graph which build by our model

have sentence and company name node. After building the meta-graphs, each graph will be defined as a unique meta-path over the structure. The semantic meaning of each meta-path is as below:

- $SS \dots S$ : Extract the sentence which is similar to other sentences.
- $KSS$ : Filter the sentence which is similar to sentence with company name.
- $CSS$ : Find out the sentence which is similar to sentence with negative keyword.

Each vertex  $S_i$  in each meta-path will have a value after random walk iteration. Therefore, vertex  $S_i$  in our task will have three value from three meta-paths separately which show in Fig. 2. For meta-path  $SS \dots S$ , each  $S_i$  will implement by infinite random walk until convergence because the meta-graph based on homogeneous graph. However, for meta-path  $KSS$  and  $CSS$  which in heterogeneous graph, we need to use limit iterations to get  $S_i$  value. And then we can define value of  $S_i$  as the summation of value which generated from each meta-path as Eq. 9. Then we derive the equations by following steps. Assume the length of document  $D$  is  $n$  that means there are  $n$  sentences in the document and can be represent as  $D = (s_1, s_2, \dots, s_n)$ . Then each sentence  $s_i$  will have a significant value which stands for their importance based on previous definitions. Therefore, our goal is ranking the value of  $s_i$  in the graph and choose Top K important sentences as our output in Eq. 8.

$$important\ sentence = TopK(V(s_i)) \tag{8}$$



**Fig. 2.** Flow of sentence selection steps

where

$$\begin{aligned}
 V(s_i) &= \sum_{l=1}^M \alpha_l \cdot PCRW(s_i | \Pi_l^{1 \sim |\Pi_l|}) \\
 &= \alpha_1 \cdot PCRW(s_i | \Pi_1) + \sum_{l=2}^M \alpha_l \cdot PCRW(s_i | \Pi_l^{1 \sim |\Pi_l|})
 \end{aligned} \tag{9}$$

The former of Eq. 9 is infinite random walk implement on meta-path  $SS \dots S$ . We can see it as a kind of ranking method which based on modified TextRank algorithm and show in Eq. 10. In original TextRank, the weight of edge is presented by numbers of co-occurrence words in two sentences. However, this concept only capture the surface feature between two sentence. Therefore, our task proposed a more meaningful weight which not only consider the semantic similarity between sentence but also apply position feature of each sentence.

Because the position information has been proved to be effective in document summarization task [9, 17]. Therefore, we define the sentence which in the front position will be more important and more related to document. Besides, in order to avoid the weight  $W_{i,j}$  prefer to position feature, we use log function to smooth position score (Table 1).

$$PCRW(s_i | \Pi_1) = (1 - d) + d \cdot \sum_{s_j \in N(s_i)} \frac{W_{ij} \cdot PCRW(\Pi_{s_j})}{\sum_{s_k \in N(s_j)} W_{jk}} \tag{10}$$

where

$$\begin{aligned}
 W_{ij} &= Cos(s_i, s_j) + P(s_i, s_j) \\
 &= \frac{\sum_{k=1}^m Vec(s_i)_k Vec(s_j)_k}{\sqrt{\sum_{k=1}^m Vec(s_i)_k^2} \sqrt{\sum_{k=1}^m Vec(s_j)_k^2}} \\
 &\quad + \log \frac{1}{\frac{pos(s_i)}{n} + \frac{pos(s_j)}{n}}
 \end{aligned} \tag{11}$$

The latter of Eq. 9 is limit iteration implement on meta-paths  $KSS$  and  $CSS$ . Unlike the infinite iteration of random walk on meta-path  $SS \dots S$ , the number



**Table 1.** The notations in Eq. 11

Notation	Description
$d$	damping factor, $d \in [0, 1]$ , usually set to 0.85
$N(s_i)$	nearby sentences which next to sentence $s_i$
$Cos(s_i, s_j)$	cosine similarity value between sentence $s_i$ and $s_j$
$m$	number of dimension in sentence vector
$Vec(s_i)_k$	element in sentence vector $s_i$
$pos(s_i)$	position of sentence $s_i$ in document $D$
$n$	total # of sentences in document $D$

of *PCRW* iteration is based on the length of meta-path. In our task, *PCRW* of two meta-paths *KSS* and *CSS* will iterate two times. In Eq. 12, the *PCRW* value of sentence  $v_i$  is summation of value  $v_j$  where  $v_j$  is not only the neighbor of  $v_i$  but also the next step of  $v_i$ . Therefore, we can figure out that the importance of a node is gather from its nearby nodes.

$$PCRW(v_i | \Pi_i^{m \sim |\Pi_i|}) = \sum_{\substack{v_j \in N(v_i) \wedge \\ v_j \in T_{m+1}}} PCRW(v_j | \Pi_i^{(m+1) \sim |\Pi_i|}) \quad (12)$$

where

$$PCRW(v_i | \Pi_i^{|\Pi_i| \sim |\Pi_i|}) = 1 \quad (13)$$

and

$$\Pi_x^{1 \sim n} = T_1 \xrightarrow{r_1} T_2 \xrightarrow{r_2} \dots \xrightarrow{r_{n-1}} T_n \quad (14)$$

### 4.3 BeamSelect

*Single Document Summarization.* After ranking by multiple random walk algorithm, we pick out Top  $K$  important sentences as heading in the summary and generate complete summary sentence by sentence. In order to reduce the space and time occupied by the sentence selection in each iteration, we uses a heuristic method, Beam Search, to estimate the Top  $K$  beam paths. Beam Search is an optimization of best-first search that explores a graph by expanding the most promising node in a limited set and usually used when the solution space of the graph is large. At the beginning, we start with the Top  $K$  significant sentences in the document rather the random states. Only the most promising sentences at each iteration of the search tree are selected for further branching, while the remaining sentences will be pruned off. Where  $K$  is parameter defined by users and control the number of the output beam paths.

In our task, the most promising sentences choose by the ranking of similarity score between candidate summary and reference document. Therefore, we can control the sentence selection in each iteration to make sure end summary have capture the semantic from original text until limit length  $K$  of summary.

**Algorithm 1.** BeamSelect Algorithm**Input:**

- $Vec(title)$  : vector representation of document title
- $Vec(s_i | s_i \in S = \{s_1, s_2, \dots, s_n\})$  : sentence representation of each sentence in document
- beam size  $k$  : limit length of summary
- $TopK(s_i)$  : heading sentence in summary

**Output:**

- Max beam path : beam path with highest similarity value
- 1:  $BeamPath[0] = \{TopK(s_i)\}$
  - 2: **for**  $i = 0$  to  $k - 1$  **do**
  - 3:      $N \leftarrow \{[y, y_{i+1}] | y \in BeamPath[i], y_{i+1} \in S\}$
  - 4:      $BeamPath[i + 1] \leftarrow \{TopK(Cos(Vec(title), Vec(y))) | y \in N\}$
  - 5: **end for**
  - 6: Return  $MAX(V(y) | y \in BeamPath[k])$

*Multi-Documents Summarization.* For multi-documents, we will face the problem that how to generate the summary with diversified sentences. Because multi-documents will have the sentence which contain similar meaning. Therefore, first we implement *BeamSelect* of each document in the same topic. Then cluster the similar sentence into same group. The amount of cluster is based on the length of longest single document summary in the topic. Second, we implement classic diversity-based reordering algorithm, MMR (Maximal Margin Relevance) [2] to solve the problem of diversity. MMR is a greedy algorithm that it will add the candidate object into ranking set in each step. The candidate object is the one have the highest MMR score and the score function define as below:

$$MMR(s_i) = \arg \max_{s_i \in R \setminus S} (\lambda Sim(Q, s_i) - (1 - \lambda) \max_{s_j \in S} Sim(s_i, s_j)) \quad (15)$$

where  $R$  is the initial ranking order when ranking set  $S$  is empty. In the Eq. 15, the former decide the similarity between candidate object  $s_i$  and query  $Q$ , the latter control the different from the objects ranked above. Therefore, based on assumptions above, we can know that the first step can retain the prestige in single document summarization and the second step can keep diversity in multi-documents summarization.

## 5 Experiment

### 5.1 Data Set and Setup

In this study, we conduct experiments on benchmark data sets DUC2001<sup>1</sup> and DUC2002<sup>2</sup> which published by the Document Understanding Conference(DUC)

<sup>1</sup> <https://www-nlpir.nist.gov/projects/duc/guidelines/2001.html>.

<sup>2</sup> <https://www-nlpir.nist.gov/projects/duc/guidelines/2002.html>.

to evaluate the effectiveness of system. We can use the two data sets to evaluate automatic extractive-based summarization of a single document and multi-documents. Table 2 shows the size and summary length of each data set above. In addition, we implement in Chinese data sets as below. The first one is LCSTS<sup>3</sup> [5] which stands for Large-scale Chinese Short Text Summarization dataset constructed from the Chinese microblogging website SinaWeibo<sup>4</sup>. We use Part III which contains 1,106 human labeled pairs (short text, summary).

Because our task focuses on financial issue, so negative events to a company may be the target of users intention. Therefore, the second Chinese data set is crawling news from multiple news sites by ourselves. The negative keywords which provided by the experts of bank auditor will be the query word in crawling step. The amount of negative keywords is 108 in 5 categories. We randomly select 20 keywords according to their proportions from 5 categories show in Table 4. For each keyword, we crawl news based on different categories in Table 3 and the total number of data set is 200. Then the reference summary of each news and the quality score of candidate summary are provided by bank auditor experts. Therefore, we can say that our collecting data set is meaningful.

**Word Model Setup.** For feature extraction in English data set, pre-trained word representation<sup>5</sup> [14] are used to initialize sentence representation. However, we need to train word model for Chinese data sets. In addition, the more consistent the data used for training, the more accurate the word model. Therefore, based on our financial task, we use query word not only contain negative keyword but also public company name which got from TEJ<sup>6</sup> (Taiwan Economic Journal). Table 4 show the keyword amount and news amount in each category. However, we don't list the news amount in each company here because the space limitations. Then we use total 797 thousands news data to train our model by CBOV model which contains 300-dimension vector.

**Table 2.** Statistics of DUC datasets

Statistic	Dataset	
	DUC2001	DUC2002
Cluster #	30	59
Document #	309	567
Task1 Limitation	100 words	100 words
Task2 Limitation	100 words	100 words

**Table 3.** Description of each feature

Category	Statistic	
	Document #	Words #
Single Short doc	2	> 600
Single Long doc	2	< 400
Single Medium doc	2	400–600
Multi doc	4	

<sup>3</sup> <http://icrc.hitsz.edu.cn/Article/show/139.html>.

<sup>4</sup> <http://www.sina.com.cn/>.

<sup>5</sup> <https://code.google.com/archive/p/word2vec/>.

<sup>6</sup> <https://www.tej.com.tw/twsite/>.

**Table 4.** The amount of negative keywords and news in each category

Keyword category	Statistic		
	Keyword #		News #
	Exp	Total	
Credit History	3	13	59497
Operate Condition	10	61	292096
Business Prospect	1	8	26185
Penalization Record	3	12	147016
Operator Background	3	14	17141

## 5.2 Evaluation Metric

We use ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [8] toolkit (version 1.5.5) as our measurement. It has been widely used in DUC task for evaluating automatic summarization in Natural Language Processing. ROUGE measures summary quality through counting overlapping units such as  $N$ -gram, word sequences and word pairs between the system generated summary (candidate summary) and gold standard (human label reference summary). Here we provide the average F-measure scores of three evaluation methods ROUGE-1, ROUGE-2 and ROUGE-L. The first two metrics based on uni-gram match and bi-gram match respectively and the third one use LCS (Longest Common Subsequence) matrix. Formally, ROUGE-1 and ROUGE-2 can represent as ROUGE- $N$  show as below:

$$ROUGE - N = \frac{\sum_{S \in \{RefSum\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{RefSum\}} \sum_{gram_n \in S} Count(gram_n)} \quad (16)$$

Where  $n$  stands for the length of the  $n$ -gram,  $gram_n$ , and  $Count_{match}(gram_n)$  is the maximum number of  $n$ -grams co-occurring in a candidate summary and reference summaries.

Similar to ROUGE- $N$ , LCS-based F-measure computed the similarity between reference summary  $X$  of length  $m$  and candidate summary  $Y$  of length  $n$  as follow:

$$ROUGE - L = \frac{(1 + \beta^2)R_{lcs}P_{lcs}}{R_{lcs} + \beta^2P_{lcs}} \quad (17)$$

where

$$R_{lcs} = \frac{LCS(X, Y)}{m}, P_{lcs} = \frac{LCS(X, Y)}{n} \quad (18)$$

$LCS(X, Y)$  is the length of a longest common subsequence of  $X$  and  $Y$ . Based on assumption above, we can figure and notice that ROUGE-L is 1 if  $X$  as same as  $Y$  and ROUGE-L is 0 when there is no common between  $X$  and  $Y$ .

The intuition is that the longer the LCS of two summary sentences is, the more similar the two summaries are. It can also seem as a more severe standard of ROUGE-N.

### 5.3 Compared Methods

We compare our summarization system with following representative approaches which briefly describe as below:

- **Random**: randomly select sentences which in the original document to compose summaries, can seem as the baseline.
- **LexRank** [4]: a famous stochastic graph-based approach which use PageRank to measure importance of sentence and exploited with MMR greedy algorithm. Therefore, it can both balance the low redundancy and high prestige of sentence.
- **DivRank** [12]: apply the *rich-gets-richer* mechanism to increase transition probability of vertex-reinforced random walk in each step to balance the score for prestige and diversity properties of sentences
- **DivSelect+CNNLM** [16]: developed an unsupervised CNN scheme to learn sentence representations, and proposed a new sentence selection algorithm DivSelect which based on PageRank to balance sentence prestige and diversity like **LexRank**. In the following sections, we use *DS - CNN* to present *DivSelect + CNNLM*.
- **EV** [3]: a novel unsupervised paragraph embedding model, which aims at not only distilling the most representative information from a paragraph but also excluding the general background information to produce a more informative low-dimensional vector representation for the paragraph of interest.

### 5.4 Experiment Results and Analysis

*ROUGE Score Results.* ROUGE toolkit can generate three scores (recall, precision and F-measure) for each evaluation. In our study, we use F-measure to evaluate different methods. The single document summarization results perform on DUC2001 and DUC2002 data sets are concluded in Tables 5 and 6.

**Table 5.** F-measure: DUC2001 Task 1

Algorithm	Rouge-1	Rouge-2	Rouge-L
Random	0.37772	0.12511	0.22197
LexRank	0.40217	0.14962	0.22506
DivRank	0.38352	0.13613	0.22210
DS-CNN	0.39577	<b>0.20060</b>	<b>0.28619</b>
EV	0.39505	0.14707	0.23888
MY	<b>0.41299</b>	0.15315	0.24431

**Table 6.** F-measure: DUC2002 Task 1

Algorithm	Rouge-1	Rouge-2	Rouge-L
Random	0.38921	0.14645	0.23504
LexRank	0.42573	0.17833	0.23983
DivRank	0.40633	0.15912	0.23109
DS-CNN	0.41272	0.16903	0.24576
EV	0.4208	0.1751	0.26386
MY	<b>0.43979</b>	<b>0.18039</b>	<b>0.26814</b>

**Table 7.** F-measure on LCSTS

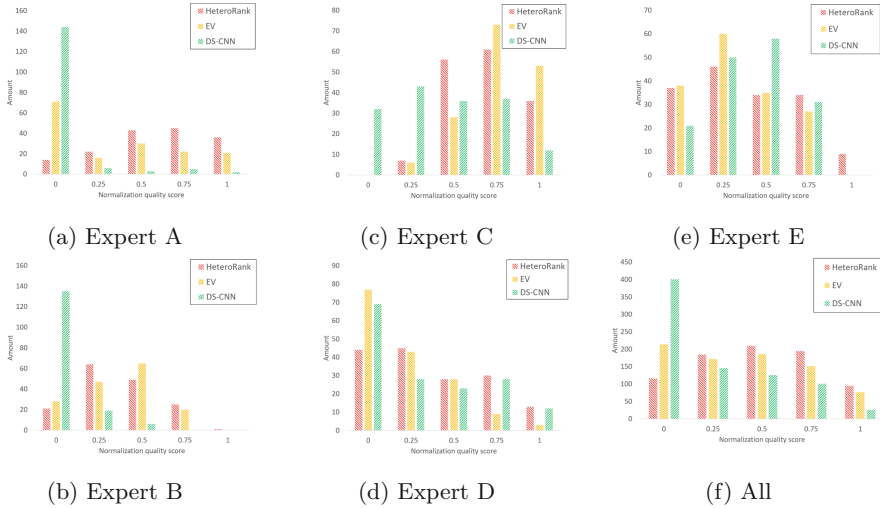
Algorithm	Rouge-1	Rouge-2	Rouge-L
Random	0.13596	0.03704	0.11833
LexRank	0.12083	0.03203	0.10662
DivRank	0.14210	0.03807	0.12262
DS-CNN	0.16056	0.04801	0.13669
EV	0.1687	0.05606	0.13399
MY	<b>0.18846</b>	<b>0.05675</b>	<b>0.16297</b>

**Table 8.** F-measure on financial data

Algorithm	Rouge-1	Rouge-2	Rouge-L
Random	0.42285	0.21218	0.27901
LexRank	0.41647	0.20567	0.23072
DivRank	0.44035	0.22836	0.24523
DS-CNN	0.47566	0.24223	0.2471
EV	0.547252	0.380844	0.445388
MY	<b>0.59022</b>	<b>0.50477</b>	<b>0.56122</b>

**Single Document Summarization.** The result clearly shown that randomly choose have poorest performance because it did not consider any ranking and diversity features. Note that the graph-based method *LexRank* and *DivRank*, both based on concept of discover truly prestigious and no redundancy sentence. *DivRank* use *rich-get-richer* mechanism however this mechanism may deviate from the centroid meaning of summary when choose the wrong seeds node in the initial. Therefore, simple classic *LexRank* performs better than *DivRank*. Apparently, our method perform the best in ROUGE-1 followed by *EV* and *DS - CNN*. Where *DS - CNN* have the highest ROUGE-2 and ROUGE-L in DUC2001, however, based the opinion proposed from [10], we can know uni-gram match score (ROUGE-1) has been proved that is most agree to human judgment. Because uni-gram co-occurrence statistics consistently correlated highly with human assessments and had high recall and precision in significance test with manual evaluation results. Therefore, we can say our approach is more convincing than *DS - CNN*. Besides, *DS - CNN* and *EV* are similar and competitive to our work. The former add new sentence which could improve the value of objective function in each step, the latter both consider density and divergence of each sentence. Whereas, our method keep the relation between sentence and user intention rather than the prestige of the original sentences only. Under this framework, we apply the title of news as user intention in experiment and show the promising result compared to others.

The result of experiment in third and fourth data sets are list in Tables 7 and 8. For experiment in LCSTS data set, the feature of user intention use the head sentence in the document to express. Because the data set didn't offer document title or other keyword. As shown by the highest ROUGE score, we can prove that the feature of sentence position will affect the quality of summary result indeed. Because our task based on financial issue, so we collect relevant news which is meaningful to bank auditor experts and experiment with different document category which show in Table 3. Therefore, please notice that the value list in Table 8 is the average score of four categories. Both company name and negative keywords which correspond with news be used as user intention in the experiment. Obviously, we can conclude that our system achieve the considerable progress than other methods.



**Fig. 3.** Score distribution of financial data set by experts.

**Multi-documents Summarization.** In this section, Table 9 show the result on DUC2001 and Table 10 present the outcome on DUC2002. Based on these experiment above, we can observe that our *HeteroRank* have higher ROUGE score than other methods. Because we not only retain the prestige in single document but also keep diversity in multi-documents.

**Table 9.** F-measure: DUC2001 Task 2

Algorithm	Rouge-1	Rouge-2	Rouge-L
Random	0.29749	0.07595	0.17621
LexRank	0.34684	0.07425	0.17618
DivRank	0.3553	0.08083	0.1832
DS-CNN	0.38245	0.11808	0.215
EV	0.33269	0.0713	0.17017
MY	<b>0.38598</b>	<b>0.12026</b>	<b>0.216</b>

**Table 10.** F-measure: DUC2002 Task 2

Algorithm	Rouge-1	Rouge-2	Rouge-L
Random	0.29914	0.05032	0.15337
LexRank	0.35803	0.08152	0.18099
DivRank	0.37335	0.08336	0.17571
DS-CNN	0.38213	0.09817	0.18941
EV	0.36325	0.09151	0.18951
MY	<b>0.38446</b>	<b>0.10175</b>	<b>0.19234</b>

*Summary Quality Score Result.* Then we compare summary quality between *HeteroRank*, *EV* and *DS – CNN* in financial data set. We ask for five bank auditor experts to evaluate each summary which generate by three methods mention above. The range of evaluate score between 1 to 5 (the higher score the better quality). Before use collecting score, we need to normalize it first. Removing the limit of data and converting it into a pure value can facilitate the comparison and weighting of indicators of different units or magnitudes. The most typical normalization is to map the data into [0,1] interval.

In Fig. 3, we can know our summary quality score concentrate in range 0.25 to 0.75 by every experts and present the normal distribution. In contrast to *HeteroRank*, *EV* and *DS – CNN* method have low quality because the score almost centralize in 0 and the amount decrease with the score increase. Besides, the average score of *HeteroRank* is 0.4903 which higher than *EV* 0.4078 and *DS – CNN* 0.25155. It means our method is more promising and meet requirements. Furthermore, we also compare the summary quality in different document category and different keyword category. Please notice that we will generate one summary by two documents in multiple summarization in our work. The summary amount and average quality score of different document category show in Table 11. Based on well perform of quality score, we can say that our method is suit for every different type document. Moreover, our quality score is even 1 more than another competitor. Table 12 illustrate the viewpoint of different keyword category, we can found that the summary in *Credit History* category has outstanding performance score. According to observation, we found that average amount of training news in category *Operate Condition*, *Penalization Record* and *Operator Background* is less than *Credit History*. However, the average amount of *Business Prospect* is much more than *Credit History*. Therefore, we can know data amount is a significant feature in training progress based on this situation. In the comparison between *HeteroRank* and *DS – CNN*, it is distinctly that our method still have the higher quality than competitor.

**Table 11.** Average quality score of different document categories.

Category	Statistic			
	Sum #	Average score		
		HR	EV	DS-CNN
Single Short sum	40	<b>0.4875</b>	0.42125	0.2875
Single Long sum	40	<b>0.55625</b>	0.5275	0.32875
Single Medium sum	40	<b>0.45125</b>	0.36125	0.21375
Multi sum	40	<b>0.46625</b>	0.32125	0.205

**Table 12.** Average quality score of different keyword categories.

Category	Statistic			
	Sum #	Average score		
		HR	EV	DS-CNN
Credit History	24	<b>0.6375</b>	0.42708	0.35208
Operate Condition	80	<b>0.48875</b>	0.41125	0.2175
Business Prospect	8	<b>0.43125</b>	0.36875	0.2125
Penalization Record	24	<b>0.4584</b>	0.37917	0.28125
Operator Background	24	0.4	<b>0.41875</b>	0.24792



## 6 Conclusions

With our proposed methods HeteroRank which used to deal textual unit ranking, and BeamSelect which used to optimize textual unit selection, we can generate summary based on user intention and still keep semantic meaning from original document. The experiment results of our model in ROUGE score show the major improvement on multiple data sets. Besides, the summary quality score also perform better than competitor. Therefore, we can say HeteroRank is promising and dependable in document summarization task.

**Acknowledgments.** The authors are grateful to SinoPac Financial Holdings Company Limited for providing the financial insights and testing data set used in this study.

## References

1. Abdi, A., Shamsuddin, S.M., Aliguliyev, R.M.: QMOS: query-based multi-documents opinion-oriented summarization. *Inf. Process. Manage.* **54**(2), 318–338 (2018)
2. Carbonell, J.G., Goldstein, J.: The use of MMR, diversity-based reranking for reordering documents and producing summaries. In: *Research and Development in Information Retrieval*, pp. 335–336 (1998)
3. Chen, K.Y., Liu, S.H., Chen, B., Wang, H.M.: An information distillation framework for extractive summarization. *IEEE/ACM Trans. Audio Speech Lang. Process.* **26**(1), 161–170 (2018). <http://dblp.uni-trier.de/db/journals/taslp/taslp26.html#ChenLCW18>
4. Erkan, G., Radev, D.R.: LexRank: graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res.* **22**, 457–479 (2004)
5. Hu, B., Chen, Q., Zhu, F.: LCSTS: a large scale Chinese short text summarization dataset. *CoRR abs/1506.05865* (2015), <http://arxiv.org/abs/1506.05865>
6. Hu, Y.H., Chen, Y.L., Chou, H.L.: Opinion mining from online hotel reviews—a text summarization approach. *Inf. Process. Manage.* **53**(2), 436–449 (2017)
7. Jung, W., Ko, Y., Seo, J.: Automatic text summarization using two-step sentence extraction. In: Myaeng, S.H., Zhou, M., Wong, K.-F., Zhang, H.-J. (eds.) *AIRS 2004. LNCS, vol. 3411*, pp. 71–81. Springer, Heidelberg (2005). <https://doi.org/10.1007/978-3-540-31871-2.7>
8. Lin, C.Y.: Rouge: a package for automatic evaluation of summaries. In: *Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)* (2004)
9. Lin, C.Y., Hovy, E.H.: Identifying topics by position. In: *Proceedings of 5th Conference on Applied Natural Language Processing*. Washington D.C., March 1997
10. Lin, C.Y., Hovy, E.: Automatic evaluation of summaries using n-gram co-occurrence statistics. In: *Proceedings of the HLT-NAACL*, 8 p. (2003). <http://research.microsoft.com/~cyl/download/papers/NAACL2003.pdf>
11. Lloret, E., Boldrini, E., Vodolazova, T., Martínez-Barco, P., Muñoz, R., Palomar, M.: A novel concept-level approach for ultra-concise opinion summarization. *Expert Syst. Appl.* **42**(20), 7148–7156 (2015)
12. Mei, Q., Guo, J., Radev, D.R.: Divrank: the interplay of prestige and diversity in information networks. In: Rao, B., Krishnapuram, B., Tomkins, A., Yang, Q. (eds.) *KDD*, pp. 1009–1018. ACM (2010). <http://dblp.uni-trier.de/db/conf/kdd/kdd2010.html#MeiGR10>

13. Mihalcea, R., Tarau, P.: TextRank: bringing order into texts. In: Proceedings of EMNLP-04 and the 2004 Conference on Empirical Methods in Natural Language Processing, July 2004
14. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS, pp. 3111–3119. Curran Associates, Inc. (2013). <http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf>
15. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: bringing order to the web. In: Proceedings of the 7th International World Wide Web Conference, Brisbane, Australia, pp. 161–172 (1998), [citeseer.nj.nec.com/page98pagerank.html](http://citeseer.nj.nec.com/page98pagerank.html)
16. Yin, W., Pei, Y.: Optimizing sentence modeling and selection for document summarization. In: Yang, Q., Wooldridge, M. (eds.) IJCAI, pp. 1383–1389. AAAI Press (2015). <http://dblp.uni-trier.de/db/conf/ijcai/ijcai2015.html#YinP15>
17. You, O., Li, W., Lu, Q., Zhang, R.: A study on position information in document summarization. In: Huang, C.R., Jurafsky, D. (eds.) COLING (Posters), pp. 919–927. Chinese Information Processing Society of China (2010). <http://dblp.uni-trier.de/db/conf/coling/coling2010p.html#YouLLZ10>

**Spatial**



# A Hierarchical Index Structure for Region-Aware Spatial Keyword Search with Edit Distance Constraint

Junye Yang<sup>1</sup>, Yong Zhang<sup>1(✉)</sup>, Huiqi Hu<sup>2</sup>, and Chunxiao Xing<sup>1</sup>

<sup>1</sup> RIIT, TNList, Department of Computer Science and Technology,  
Tsinghua University, Beijing, China

yjy17@mails.tsinghua.edu.cn, {zhangyong05,xingcx}@tsinghua.edu.cn

<sup>2</sup> Institute for Data Science and Engineering, East China Normal University,  
Shanghai, China

hquh@sei.ecnu.edu.cn

**Abstract.** Location-based services have become widely available on a variety of devices. Due to the errors in user input as well as geo-textual databases, supporting error-tolerant spatio-textual search becomes an important problem in the field of spatial keyword search. Edit distance is the most widely used metrics to capture typographical errors. However, existing techniques for spatio-textual similarity query mainly focused on the set based textual relevance, but they cannot work well for edit distance due to the lack of filter power, which would involve larger overhead of computing edit distance. In this paper, we propose a novel framework to solve the region aware top- $k$  similarity search problem with edit distance constraint. We first propose a hierarchical index structure to capture signatures of both spatial and textual relevance. We then utilize the prefix filter techniques to support top- $k$  similarity search on the index. We further propose an estimation based method and a greedy search algorithm to make full use of the filter power of the hierarchical index. Experimental results on real world POI datasets show that our method outperforms state-of-the-art methods by up to two orders of magnitude.

## 1 Introduction

With the growing popularity of mobile devices, location-based services have been widely deployed and well accepted. The user generated data (UGD) of such services can actively contribute to the location based social media. However, it is not easy to retrieve them by simply matching especially when they are not fully cleaned and standardized. Error-tolerant search is a technique to reduce the gap between user intention behind its query and their required data, it is very necessary for spatio-textual query over UGD with two-fold reasons: on the one hand, there exist lots of misspellings. Typos appear frequently in both UGD and user's typing queries when users are utilizing smartphones for input. On the other hand, users may use different spelling variants for the same object. For instance, both of "harbour" and "harbor" are acceptable to users due to different

spelling habits. Supporting error-tolerant search allows us to better overcome the limitation of data description and make spatio-textual query more user friendly.

In this paper, we study the problem of error-tolerant spatio-textual search. For the ease of user’s query, we support boolean range query [4], which allow users to use “zoom in”, “zoom out” or “shift” in map application to adjust their interesting locations. To this end, we need to let users specify a region they are interested in. As edit distance is the most widely used metrics to capture typographical errors [13], we target at the problem of region-aware top- $k$  similarity search with edit distance constraint. Given a spatial region and a textual description, it finds the  $k$  most related spatio-textual objects contained in the region with minimum edit distances. For example, when the textual input is “restaurant”, the user probably aims at finding some nearby restaurants. No matter what the selected region is, the query always display the  $k$  objects within the region that have smallest edit distances with “restaurant”, which including both objects named correctly with “restaurants” and objects that have misspelling (e.g. “resturant”) within the dataset.

Many studies have investigated top- $k$  string similarity search problem with edit distance constraint, which have yielded a batch of effective techniques [15, 17, 19]. But they do not involve any spatial information of data and query. Therefore they cannot support our problem. On the other hand, spatio-textual similarity search have attracted significant attentions from recent work. Many studies [4, 10, 11, 21] follow Cong et al. [6], a pioneering study of top- $k$  spatio-textual query, which joint measures spatial and textual similarity. But they mainly focus on set-based textual relevance, i.e. the text of an object is represented as a set of tokens and some similarity functions (e.g. Jaccard similarity [3]) are used to measure the relevance. However, none of current solutions can efficiently satisfy the region query with top- $k$  edit distance. Compared with set-based textual relevance which costs  $\mathcal{O}(n)$  time for verification, computing edit distance is prohibitively expensive for large databases since the price of verification is  $\mathcal{O}(n^2)$ . As the previous studies measure textual relevance based on either exact keyword matching or set-based metrics, they cannot perform well for edit distance due to the limited filter power. There is a need for more powerful pruning techniques to avoid the expensive verification. Besides, another challenging problem is how to smartly combine spatial and textual filters so as to minimize the number of verifications without heavy space and time overhead.

To address this problem, we firstly try to combine the spatial index (e.g. R-Tree) with the prefix filtering technique in string similarity query and devise a hierarchical index, where inverted lists of  $q$ -grams are attached into the nodes of spatial index. We have the observation that grams with different length have different filtering power [8]. Thus we utilize variable lengths of  $q$ -grams in different levels of the index to facilitate pruning for top- $k$  query and avoid duplicate verifications. To this end, we introduce the concept of *filter power* and *filter ability*. With respect to the variable lengths of grams and prefix filtering, strong filter power enables us to find the most relevant results efficiently within given threshold. However, it also limits the filter ability if we cannot find the exact

top- $k$  results within the threshold. In this case, we have to verify all the rest objects one by one, which in turn involves many verifications. With the help of hierarchical index, we can avoid such computation using the spatial information. The proposed top- $k$  algorithm takes full advantage of spatial and textual pruning to generate small set of candidates. Further more, we propose two optimizations to make further use of the filter power of the hierarchical index. In summary, the main contributions of the paper are as follows:

- We propose a hierarchical index to solve the region-aware top- $k$  error-tolerant search problem with edit distance constraint. To the best of our knowledge, this is the first study on this problem.
- We design an efficient top- $k$  searching algorithm that takes advantage of both spatial and textual filtering based on the index.
- We further propose an estimation based method and a greedy search algorithm to improve the efficiency.
- We conduct experiments on real world datasets. Experimental results show our method outperforms state-of-the-art methods by up to two orders of magnitude.

The rest of the paper is organized as follows. We formulate the problem and review the related works in Sect. 2. The hierarchical index and its construction are proposed in Sect. 3. We propose the top- $k$  searching algorithm in Sect. 4 and its optimizations in Sect. 5. Experimental results are reported in Sect. 6. Conclusions are made in Sect. 7.

## 2 Preliminary

### 2.1 Problem Definition

Let  $\mathcal{S}$  be a collection of geo-textual data. Each object  $o \in \mathcal{S}$  is defined as a pair  $(o.pos, o.text)$ . Here  $pos$  is a location descriptor in multidimensional space and  $text$  is a string that describes the object. Given two strings  $r$  and  $s$ , the edit distance between  $r$  and  $s$ , denoted as  $ED(r, s)$ , is the minimum number of edit operations (including substitution, insertion, and deletion) needed to transform  $r$  to  $s$ . Next we formulate the problem of top- $k$  region aware similarity search with edit distance constraint.

**Definition 1** (*Top- $k$  Region-aware Similarity Search*). *Given a collection of geo-textual objects  $\mathcal{S}$ , a region  $\mathcal{M}$ , a query string  $s$  and a number  $k$ , the region aware top- $k$  similarity search problem aims at finding a subset of the geo-textual objects  $\mathcal{H}$  s.t.  $|\mathcal{H}| = k$ , and for any  $o \in \mathcal{H}$ ,  $o.pos \in \mathcal{M}$  and  $o' \in \mathcal{S} - \mathcal{H}$ ,  $o'.pos \in \mathcal{M}$ , we have  $ED(o.text, s) \leq ED(o'.text, s)$ .*

*Example 1.* Figure 1 shows 10 geo-textual objects. Given a query string  $s = \text{“harbuor”}$ , a region  $\mathcal{M} = \{(7, 8), (29, 26)\}$  and  $k = 2$ , the edit distances between query string and ten objects are: 1, 3, 2, 3, 3, 3, 3, 2, 3, 3, respectively. Because  $o_6, o_8, o_9$  and  $o_{10}$  are not located in the  $\mathcal{M}$ , they will not be the answers.  $ED(o_1, s) = 1$  and  $ED(o_3, s) = 2$  are two smallest ones among all other 7 objects, thus the final answer is  $\{o_1, o_3\}$ .

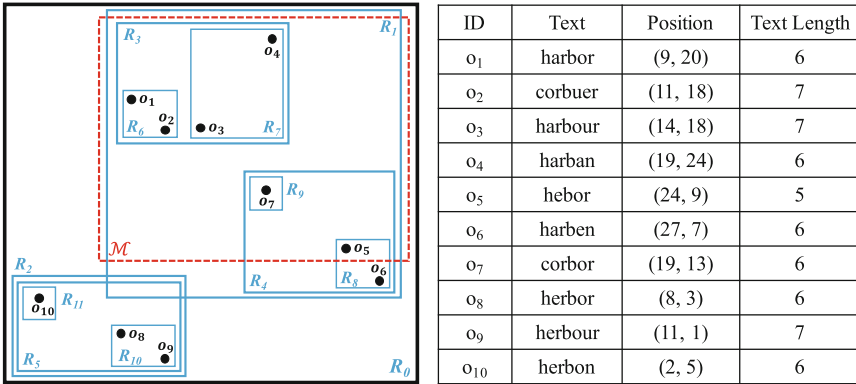


Fig. 1. A running example

## 2.2 Related Work

To the best of our knowledge, we are the first study on the spatio-textual query for top- $k$  requirement of edit distance. The only work that supports spatio-textual similarity search with constraint on edit distance is Yao et al. [20], which studies the problem of approximate string search in spatial database. They find similar results where the edit distance between the string query and object must be within a pre-defined threshold. However, they can only approximately answer matched results, where false negatives may exist. Besides, they only support threshold based similarity search. The problem is different from ours as we try to answer exact top- $k$  query without false negatives.

**Geo-Text Query Processing.** There are many studies on spatio-textual query processing [4, 6, 7, 11]. Cong et al. [5] has provided a comprehensive field introduction of current works. Also Chen et al. [4] give an all-around experimental survey of 12 state-of-the-art geo-textual indices with compared spatio-textual queries. A spatio-textual query usually includes information for both spatial and textual requirement. For the spatial dimension, region query (which finds all objects within a given region) and distance based query (which measures query and object by euclidean distance) are two common queries. For the textual dimension, the texts of objects are usually represented as a set of tokens and the set based query semantics are commonly adopted. For examples, Chen et al. [3] focus on the “AND” and “OR” semantics on token sets to evaluate textual relevancy. Li et al. [10] only return objects that all the tokens of the query are included in the token set of the object. A general solution to answer those geo-text query is to integrate textual information into some spatial indexes (e.g. R-Tree, quadtree) and then leverage the hybrid index to filter out irrelevant objects. Instead of modeling text as a set of tokens, we focus on region query and edit distance where the textual relevance is different and more complicated. We also provided how to address the problem by integrating techniques

of string similarity search into spatial indexes and developing efficient querying algorithms on them.

**Top-k Spatio-Textual Search.** One typical geo-text query is to find top- $k$  objects with maximum spatio-textual proximity. It has received significant attention and some studies focused on the top- $k$  spatio-textual search [6, 11, 14, 21, 22]. They try to find top- $k$  most relevant objects by considering a normalized similarity function containing both spatial proximity and textual relevancy, where the textual relevance are still set based similarity functions. To address the problem, Cong et al. [6] combine the inverted lists with R-Trees and proposed the IR-Tree to compute top- $k$  answers. Zhang et al. [21] combine inverted index and Quadtree.

**String Similarity Search.** String similarity search algorithms aim to find all the similar strings within a given threshold [9, 16, 18, 23]. There are also many previous studies for top- $k$  string similarity query, such as [15, 17, 19, 24]. However, Those above works only consider textual queries while we need to consider both spatial and textual requirements.

### 3 The Hierarchical Tree Index

In this section, we propose a hierarchical index structure to joint index the spatial and text information. We first discuss how to extend the prefix filtering technique to support top- $k$  string similarity search in Sect. 3.1. Then we propose the algorithm for index construction in Sect. 3.2.

#### 3.1 Support Top-k Similarity Search with Prefix Filter

Prefix Filter [2] is a state-of-the-art technique for threshold-based string similarity search. Let  $|r|$  denote the length of  $r$ ,  $Q(r)$  denote the set of  $q$ -grams of string  $r$  and  $q_i^r$  is the  $i^{th}$  gram in  $Q(r)$ . The basic idea is to first fix a global ordering on the  $q$ -grams of all the strings. Then we sort all the  $q$ -grams according to a global order and use  $\mathcal{P}_q^\tau(r)$  to denote the  $\tau$ -prefix of string  $Q(r)$ . Specifically, as one edit operation can destroy at most  $q$  grams, given a edit distance threshold  $\tau$  we have  $|\mathcal{P}_q^\tau(r)| = q\tau + 1$ . Then we can filter out dissimilar strings using Lemma 1.

**Lemma 1** (*Prefix Filter [2]*). *Given two strings  $s, r$  and an edit distance threshold  $\tau$ , if  $|\mathcal{P}_q^\tau(r) \cap \mathcal{P}_q^\tau(s)| = \emptyset$ , then we have  $ED(r, s) > \tau$ .*

Then the problem becomes how to extend the prefix filter to support top- $k$  similarity search. The question here is before answering top- $k$  search, we have no idea of which threshold to be specified. So the index should be able to deal with any threshold. Nevertheless, we have the following observations of prefix filter which can help us construct such an index:

- Strings with the same length have the  $\tau$ -prefix with same length i.e.  $q\tau + 1$ .
- For any given string  $r$ , we have  $\mathcal{P}_{\tau-1}^q(r) \subseteq \mathcal{P}_\tau^q(r)$ .



Thus we can group strings by length and construct inverted index for each group. Let us denote the group with length  $l$  as  $\mathcal{D}_l$ . For every string  $r$ , we add  $Q(r)$  into the corresponding inverted lists in  $\mathcal{D}_{|r|}$ . The entry of each inverted list is a  $q$ -gram and each item in the inverted list is in the format of  $\langle id, pos \rangle$ , where  $id$  is the id of object and  $pos$  is the position among all tokens in that object according to the global ordering. Given a query  $s$ , we enumerate  $\tau$  incrementally from 0 and look at the  $\tau$ -prefix of  $s$ . Given a threshold  $\tau$ , for all strings  $r \in \mathcal{D}_l$ , we can just look at whether there are overlapping between the  $\tau$ -prefix of  $s$  and that of  $r$  by traversing the inverted lists.

### 3.2 Index Construction

The top- $k$  region aware similarity search calls for the query processing of both the text relevance and the spatial constraint. As is known to all, R-Tree is the dominant spatial index structure which can efficiently support boolean region search. For string similarity search, we slice the string collection into  $q$ -grams and construct inverted list for them.

Despite the efficiency of R-Tree and inverted list, the implementation needs further optimization. The intuition is based on the search order: for boolean range query, we need to traverse the R-Tree in a top-down manner from root to leaf nodes; for top- $k$  string search, we need to enumerate from small to large thresholds. Then a chance for improvement is that we can filter out dissimilar objects in the process of boolean region query. Thus, it calls for a hierarchical index with which we can do pruning with the text similarity while traversing the non-leaf nodes of the R-Tree. To this end, we propose a hierarchical index that has inverted lists attached to every node.

This hierarchical index is essentially a variant of R-Tree. It can be constructed iteratively in a top down manner by attaching inverted lists to each node for the objects contained in the sub-tree rooted at the it. But here comes the problem about how to allocate the inverted lists to different levels of nodes. A simple idea is to just build the inverted lists for the objects in each node. However, as objects in a non-leaf node can be obtained from all of its children, this method could lead to large space overhead. We have an observation that for a given collection of strings, larger value of  $q$  would result in smaller size of inverted lists. Inspired by this, we use large value of  $q$  in higher levels and small value of  $q$  in lower levels as nodes in higher levels contain more objects. Besides, we can also take advantage of such index structure to improve the performance of top- $k$  search. The details will be discussed later in Sect. 4.

Algorithm 1 shows the process of constructing the hierarchical index. We first initialize  $\mathcal{HT}$  by constructing the R-Tree with spatial information of all the geo-textual objects (line 2). We then traverse the R-Tree in a top down manner to construct inverted lists for each node. We select the size of  $q$  i.e. the length of grams for each level empirically: suppose the leaf nodes are in level 1, the value of  $q$  in level  $i$  will be set as  $i + 1$  (line 4). In each node  $N$ , we first group all the strings by length (line 5). Then we construct the inverted lists  $\mathcal{L}_l^N$  for each group  $\mathcal{D}_l$  in a node  $N$  (line 7). Figure 2 illustrates the index structure.

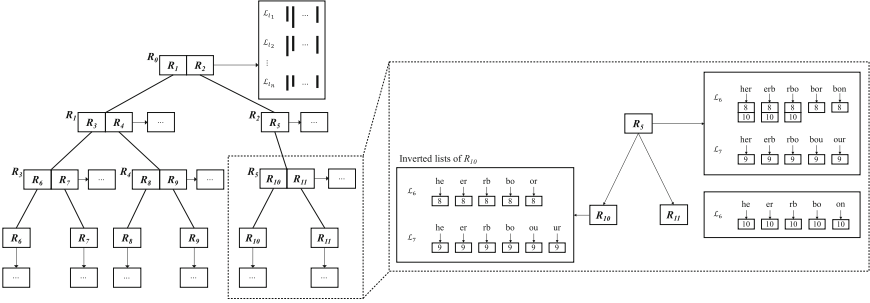


Fig. 2. The hierarchical index

### 4 The Basic Top-k Search Algorithm

In this section, we discuss how to perform region aware top- $k$  error-tolerant search on the hierarchical index proposed in Sect. 3. The basic idea is as following: we maintain a priority queue  $\mathcal{H}$  to keep the current  $k$  promising results. Let  $UB^{\mathcal{H}}$  denote the largest edit distance between the strings in  $\mathcal{H}$  to the query  $s$ . Obvious  $UB^{\mathcal{H}}$  is an upper bound of the edit distances of top- $k$  results to the query. In other words, we can prune an object if its edit distance to the query is no smaller than  $UB^{\mathcal{H}}$ , which can be estimated with the prefix filter.

---

**Algorithm 1.** Index Construction Algorithm( $S$ )

---

```

Input:  $S$ : The collection of geo-textual objects
Output:  $\mathcal{HT}$ : The Hierarchical Index of geo-textual objects
1 begin
2   Initialize  $\mathcal{HT}$  by constructing an R-Tree with only spatial information;
3   foreach node  $N \in \mathcal{HT}$  do
4      $q_i = N.level + 1$ ;
5     Group objects within  $N$  by string length  $l$  into  $\mathcal{D}_l$ ;
6     foreach group  $\mathcal{D}_l$  do
7       Construct the inverted lists  $\mathcal{L}_i^N$  with  $q_i$ -grams;
8   return  $\mathcal{HT}$ ;
9 end

```

---

Unlike threshold based similarity search, the top- $k$  search problem concentrates on not only filter power, but also *filter ability*. Given two strings, the filter ability is the maximum threshold that a filter can support. Suppose the gram length is  $q$ , the length of two strings are  $a$  and  $b$  respectively, we denote the filter ability as  $\mathcal{F}_q(a, b)$ . Given two strings  $s$  and  $r$ , the filter ability is calculated as following.

**Lemma 2.** Given two strings  $s$  and  $r$ , the filter ability of prefix filter is  $\mathcal{F}_q(|s|, |r|) = \lfloor \frac{\min(|r|, |s|) - q}{q} \rfloor$ , where  $q$  is the length of  $q$ -grams.

For instance, we have  $|s| = 7$ ,  $|r| = 8$  and  $q = 3$ , then the filter ability will be  $\lfloor \frac{7-3}{3} \rfloor = 1$ . It means that we can only compute candidates for  $\tau = 0, 1$ . If we want to compute candidates for  $\tau \geq 2$ , we need to use a smaller  $q$ .

Then we talk about how to perform top- $k$  algorithm for a given query string  $s$  and region constraint  $\mathcal{M}$  on the hierarchical index. Since in boolean region query, we search the R-Tree from root to leaf nodes while in the process of top- $k$  search we enumerate the threshold in ascending order. The most crucial problem is how to minimize the number of verifications. We observed that different  $q$ -grams make different contributions to the prefix filter. In top- $k$  search, we need to enumerate the threshold from 0 incrementally. When the threshold becomes larger than the filter ability, no string can be pruned. In this case, we need to regard every string as candidate and verify all of them with a linear scan. Here there is a trade-off between filter power and filter ability. Using the  $q$ -grams with larger value, we can leverage the strong filter power to find the most similar results at once and to reduce the value of upper bound  $\text{UB}^{\mathcal{H}}$  quickly. But at the same time, the filter ability will be lower, which means it will leaves most of the objects to be verified for linear scan if the algorithm is not terminated. Thanks to the hierarchical index structure, we can postpone this step by leveraging the spatial constraint to further prune more out-of-region objects.

Following the above route, we treat the nodes differently according to their relation with the given region  $\mathcal{M}$ . There are three situations: (1) If a node  $N$  has no overlap with  $\mathcal{M}$ , we can prune the subtree rooted by  $N$ . (2) If a non-leaf node  $N$  has overlapping with  $\mathcal{M}$  but not contained by  $\mathcal{M}$ , that means we can further utilize the region constraint to avoid linear scan in this node. Thus for each group  $\mathcal{D}_l$  in this node, we perform prefix filter to generate several candidate for the thresholds from 0 to  $\min(\text{UB}^{\mathcal{H}}, \mathcal{F}_{q_i}(l, |s|))$ . For the thresholds that are beyond the filter ability, we leave them to the children of  $N$ . (3) If a node  $N$  that is fully contained by  $\mathcal{M}$  or  $N$  is a leaf node, that means we can no longer take advantage of the region constraint to avoid scanning the strings beyond filter ability. Thus we stop looking at its children. Instead, we first perform prefix filter for all strings in  $N$  and then do linear scan on all groups beyond the filter ability.

---

**Algorithm 2.** Hierarchical Top-k Search Algorithm( $\mathcal{HT}$ ,  $\mathcal{M}$ ,  $s$ ,  $k$ )

---

**Input:**  $\mathcal{HT}$ : The Hierarchical Index of geo-textual objects,  $\mathcal{M}$ : The given region,  $s$ : The query text,  $k$ : The number of results

**Output:**  $\mathcal{H}$ : The top- $k$  results

```

1 begin
2   Initialize the heap  $\mathcal{H}$  and  $\text{UB}^{\mathcal{H}}$ ;
3   Traverse( $\mathcal{HT}.height$ ,  $\mathcal{M}$ ,  $s$ ,  $k$ ,  $\mathcal{H}$ );
4   return  $\mathcal{H}$ ;
5 end

```

---

We still use the query in Example 1 to show how the Algorithm 2 works. Suppose after the initialization,  $\mathcal{H} = \{o_5, o_7\}$ , so the upper bound  $\text{UB}^{\mathcal{H}}$  is 3. We first visit node  $R_0$  and start from  $l = 5$ , because the group for  $l = |s| - \text{UB}^{\mathcal{H}} = 4$ , i.e.,  $\mathcal{D}_4$ , is empty. Note that the  $q$  of  $R_0$  is 5, so  $\tau \in [0, \min(\text{UB}^{\mathcal{H}}, \mathcal{F}_q(l, |s|))] =$

**Algorithm 3.**  $\text{Traverse}(i, \mathcal{M}, s, k, \mathcal{H})$ 


---

```

1 begin
2   if  $i$  is 1 then
3     foreach node  $N$  in level  $i$ ,  $N.\text{region} \cap \mathcal{M} \neq \emptyset$  do
4       for  $l \in [|s| - \text{UB}^{\mathcal{H}}, |s| + \text{UB}^{\mathcal{H}}]$  do
5          $\mathcal{C}_l :=$  candidates from  $\mathcal{D}_l$  for threshold from 0 to
          min( $\text{UB}^{\mathcal{H}}, \mathcal{F}_{q_i}(l, |s|)$ );
6         foreach  $c \in \mathcal{C}_l$  do
7           Verify( $c, \mathcal{M}, s, \mathcal{H}$ );
8       foreach  $c \in N$  and  $c$  is beyond the filter ability do
9         Verify( $c, \mathcal{M}, s, \mathcal{H}$ );
10    else
11      foreach node  $N$  in level  $i$ ,  $N.\text{region} \cap \mathcal{M} \neq \emptyset$  do
12        if  $N.\text{region}$  is contained by  $\mathcal{M}$  then
13          The same with line 4 to line 9;
14        else if  $N.\text{region}$  has overlapping with  $\mathcal{M}$  then
15          The same with line 4 to line 7;
16          foreach child node  $N_c$  of  $N$  that overlaps  $\mathcal{M}$  do
17            Traverse( $i - 1, N_c, \mathcal{M}, s$ );
18 end

```

---

$\{0\}$ . In  $\mathcal{D}_5, \mathcal{D}_6$  and  $\mathcal{D}_7$ , there is no string that has common 5-gram with the prefix of  $s$ , thus we don't get any candidate in this node. Then we come to  $R_1 (q = 4)$ . For  $l = 5, \tau = 0$ , the string in  $\mathcal{D}_5 = \{o_5\}$  has no common 4-gram with  $s$ , so  $\mathcal{C}_5 = \emptyset$ . For  $l = 6, \tau = 0$  and  $l = 7, \tau = 0$ , we get two non-empty candidates set  $\mathcal{C}_6 = \{o_1, o_4, o_6\}$  and  $\mathcal{C}_7 = \{o_3\}$ . After verification process, the  $\text{UB}^{\mathcal{H}}$  is 2. Similarly, we visit  $R_3, R_4, R_8$  and  $R_9$ , but  $\mathcal{H}$  does not change. And we will not visit  $R_2$  because it doesn't overlap with  $\mathcal{M}$ . Finally, the algorithm terminates with  $\mathcal{H} = \{o_1, o_3\}$ .

## 5 Query Optimization for Internal Nodes

As inverted lists in different levels have  $q$ -grams with different length, we can take advantage of the varied filter power and ability of such levels to further improve the performance. To this end, in this section we proposed an estimation based method to decide the nodes we need to search in Sect. 5.1 and designed a light-weighted greedy algorithm to fulfill this task with minor extra overhead in Sect. 5.2.

### 5.1 Estimation Based Node Selection

Given an internal node  $N_0$  that is fully contained in the region, the intuition of estimation based method is to select nodes in the subtree rooted by  $N_0$  that have

the least cost. First of all, we need to guarantee that there is no false negative. Based on the property of R-Tree, we have an observation about the correctness as in Lemma 3.

**Lemma 3.** *Given a non-leaf node  $N_0$  in  $\mathcal{HT}$  contained by  $\mathcal{M}$ , any combination of  $N_0$ 's children that covers the region of  $N_0$  will include all the possible results of top- $k$  string similarity search.*

As the dominant cost of top- $k$  string similarity search is verification, especially verifying the strings beyond filter ability, we use the number of estimated candidates to denote the cost in this process. To this end, we adopted *min-wise hash* technique [1] to estimate the number of verifications that we need to perform for each node in the subtree rooted by  $N_0$ . The intuition is that each inverted list can be represented by a min-hash signature. And we can estimate the number of candidates for a given threshold  $\tau$  without expensive scanning on the inverted lists. The way to generate a min-hash signature is to use a uniformly distribution  $U$  to assign a random value in the range  $[0, 1]$  for each object in the list. For each inverted list  $\mathcal{L}[t_i]$  we generate  $K$  signatures<sup>1</sup> and form a feature vector denoted as  $Y_i$ . We can formulate it as below:

$$Y_i^j = \min\{U^j(o), o \in \mathcal{L}[g_i]\} \tag{1}$$

where  $Y_i^j$  is the  $j^{th}$  value of  $Y_i$  and  $g_i$  is the corresponding  $q$ -gram. Then the signature vector of inverted list  $\mathcal{L}[g_i]$  is correspondingly  $Y_i = (Y_i^1, \dots, Y_i^K)$ .

With such definition, given the node  $N_0$  and query string  $s$ , we can estimate the number of candidates within  $\tau$  edit distance from  $s$  in  $N$  by leveraging the VSol estimator [12]. The idea is as following:

Suppose

$$(Z^1, \dots, Z^K) = \left( \min_{i=1, \dots, M} Y_i^1, \dots, \min_{i=1, \dots, M} Y_i^K \right) \tag{2}$$

is the signature vector of the union  $\bigcup_i \mathcal{L}[g_i^s]$ , where  $\bigcup g_i^s = Q(s)$  and  $M$  is the number of inverted lists. Let

$$\hat{\rho} = \frac{1}{K} \sum_{j=1}^K \mathbf{1}\{\exists i_1, \dots, i_L : Y_{i_1}^j = \dots = Y_{i_L}^j = Z^j\} \tag{3}$$

be the estimator of  $\rho$ , where  $L = |s| - 1 - (\tau - 1) \cdot q$ . And the size of  $\bigcup_i \mathcal{L}[g_i^s]$  can be estimated as follows:

$$\left| \bigcup_i \mathcal{L}[g_i^s] \right| = \frac{K}{\sum_{j=1}^K (Z^j)} - 1 \tag{4}$$

Finally the number of candidates can be estimated as

$$|\widehat{\mathcal{C}}_\tau(s)| = \hat{\rho} \times \left( \frac{K}{\sum_{j=1}^K (Z^j)} - 1 \right) \tag{5}$$

<sup>1</sup> The uniformly distribution used in the  $i^{th}$  time is  $U^i$  where  $i \in [1, K]$ .

As the VSol estimator can only support estimating the cardinality for a given threshold, we extend it to the scenario of top- $k$  search in the following way. Suppose when we reach the node  $N_0$ , the lowest edit distance between elements in  $\mathcal{H}$  and  $s$  is  $LB^{\mathcal{H}}$ , we will increase  $\tau$  from  $LB^{\mathcal{H}}$  to  $UB^{\mathcal{H}}$  or until we find a threshold that have  $\frac{k}{\epsilon}$  estimated candidates. Here  $\epsilon$  is a tunable parameter which denote the ratio of candidates that can result in updating of  $\mathcal{H}$ . We set its value as 0.1 empirically. The detailed process is shown in Algorithm 4.

---

**Algorithm 4.** Candidates Estimation Algorithm( $N, s, \mathcal{H}$ )

---

**Input:**  $N$ : The node,  $s$ : The geo-textual query

**Output:** The estimated number of candidates

```

1 begin
2   for  $\tau \in [LB^{\mathcal{H}}, UB^{\mathcal{H}}]$  do
3     Estimate the value of  $|\widehat{C}_\tau(s)|$  on node  $N$  using Equation 2 to 5;
4     if  $\tau = UB^{\mathcal{H}}$  or  $|\widehat{C}_\tau(s)| \geq \frac{k}{\epsilon}$  then
5       return  $|\widehat{C}_\tau(s)|$ ;
6 end

```

---

With Algorithm 4 we are able to obtain the cost of visiting each node. Note that although the VSol estimator could result in false negative by leveraging the min-hash technique, our estimation based method can still find correct top- $k$  results. The reason is that VSol estimator utilize the min-hash technique to directly obtain the candidates, while we only use it to decide which nodes we need to traverse. According to Lemma 3, as long as selected nodes cover the region of  $N_0$ , there would be no false negative in our method.

Then the next problem becomes how to select the combination of nodes and perform filtering and verification on them. The goal is to find the nodes with minimum weight on the basis of satisfying Lemma 3. Given a subtree in which each node  $N$  has a weight  $\omega(N)$  calculated by Algorithm 4, we can reach this goal with a greedy strategy. Algorithm 5 describes the detailed process. We first traverse all the internal nodes of the subtree rooted by  $N_0$  in a bottom-up manner and decide the minimum weight each node can have (line 3–9). In this process, we calculate all children’s weight of a node (line 5) and compare the sum with its weight. Then select the smaller one as the node’s weight (line 7 and 9). Then we perform a level traverse on the subtree and obtain all the selected node in the previous step (line 10–16).

*Example 2.* Figure 3 shows an example of the nodes selection process. At first, each node has a black number, i.e.,  $\omega(N)$ . And each node’s red bold number is empty, which represents the updated value of  $\omega(N)$ , denoted as  $\omega'(N)$ . For each leaf node  $N$ ,  $\omega'(N) = \omega(N)$ . Next we go from bottom to up. For  $R_3$ ,  $3 + 4 = 7 > 5$ , thus  $\omega(R_3) = 5$  and we mark it with *selected*, denoted with red color. For  $R_4$ ,  $2 + 1 = 3 < 5$ , thus  $\omega(R_3) = 3$  and we mark its children with *selected*. Similarly, we update the weight for  $R_5$ ,  $R_1$ ,  $R_2$  and  $R_0$ . Then we have

---

**Algorithm 5.** Nodes Selection Algorithm( $N_0, s$ )

---

**Input:**  $N_0$ : Root of a subtree,  $s$ : The geo-textual query

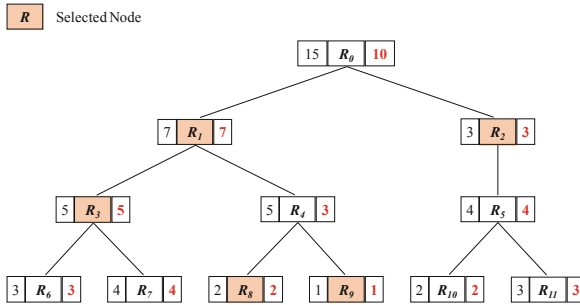
**Output:**  $\mathcal{N}$ : The selected nodes

```

1 begin
2   Let  $i_0$  be the level of  $N_0$ ;
3   for  $i = 2$  to  $i_0$  do
4     foreach node  $N$  in level  $i$  do
5        $W = \sum_{n \in \text{children of } N} \omega(n)$ ;
6       if  $W < \omega(N)$  then
7         mark all children of  $N$  as selected, set  $\omega(N) = W$ ;
8       else
9         mark  $N$  as selected;
10  Initialize  $\mathcal{N} = \emptyset$  and  $Q = N_0$ ;
11  while  $Q \neq \emptyset$  do
12    Dequeue the front node  $N_f$  from  $Q$ ;
13    if  $N_f$  is selected then
14      Insert  $N_f$  into  $\mathcal{N}$ ;
15    else
16       $Q = Q \cup \{\text{children of } N_f\}$ ;
17  return  $\mathcal{N}$ ;
18 end

```

---



**Fig. 3.** The selection of tree nodes (Color figure online)

the minimum weight 10 and 5 marked nodes. Finally we use breadth-first search to traverse the tree and push marked nodes into result. Note that if a node and its child are both marked, such as  $R_1$ , we only need add the parent to the results and prune the traversal of it. So the result is  $\mathcal{N} = \{R_1, R_2\}$ .

Finally, we talk about how to integrate the estimation based method into the top- $k$  search algorithm. This technique is applied to the scenario when a non-leaf node  $N_0$  is contained by the region. We first adopt Algorithm 4 to estimate the number of candidates for each node in the subtree rooted by  $N_0$ . We then use

Algorithm 5 to select the set  $\mathcal{N}$  of nodes to be searched. Instead of stopping at  $N_0$ , for each node  $N \in \mathcal{N}$  we perform prefix filter and then linear scan to verify all the candidates.

## 5.2 Fine-Grained Greedy Approximation

Although the estimation based method can fully utilize the varied filter power and ability in different level, it involves heavy overhead. Firstly, we need to create signatures for inverted lists in each node. Secondly, given a non-leaf node  $N_0$ , we need to compute the cost for all nodes in the subtree, which leads to a large amount of overhead.

To avoid this problem as well as take advantage of different filter power and ability, we propose a greedy approximation rather than calculate the accurate weight for each node. The intuition is that we should make full use of the filter power of inverted lists in all levels. Then we traverse the subtree in a greedy manner: for all the non-leaf nodes in the subtree, we perform prefix filter and avoid linear scan. In the leaf nodes, we first perform prefix filter and then verify all the remaining strings. In this case, as we can utilize prefix filter from all upper levels, we can avoid linear scan on as many strings beyond filter ability as possible. Based on such idea, we propose a greedy search algorithm as is shown in Algorithm 6.

# 6 Evaluation

## 6.1 Experiment Setup

We use two real world POI datasets in our experiments: Texas (TX) and California (CA) road network and streets data. They are from the [open street map project](http://www.openstreetmap.org/)<sup>2</sup>. Each point has the information of longitude and latitude coordinates as the spatial information and some strings as the textual information. We combine all the words as its associated string. Two datasets both have 1 million points. The real datasets are in two dimensions. Details of the datasets are shown in Table 1. The default value for  $k$  is 6 and default value for  $\theta$  is 1%. All the algorithms were implemented using C++ and compiled using GCC 4.9.4 with -O3 flag. All the experiments were run on a Ubuntu server machine with 2.40 GHz Intel(R) Xeon E5620 CPU with 4 cores and 64 GB memory.

Table 1. Datasets

Datasets	Cardinality	Avg Len	Max Len	Min Len
CA	1,000,000	12	56	1
TX	1,000,000	11	47	1

<sup>2</sup> <http://www.openstreetmap.org/>.



**Algorithm 6.** GreedySearch( $N, \mathcal{M}, s$ )

---

**Input:**  $N$ : The node to be searched,  $\mathcal{M}$ : The given region  
 $s$ : The geo-textual query

```

1 begin
2   // Replace line 13 in Algorithm 3 with this algorithm;
3   if  $N$  is leaf node then
4     Perform prefix filter with Algorithm 3 on  $N$  to find all the candidates;
5     for each  $c \in N$  and  $c$  is beyond the filter ability do
6       | verify( $c, \mathcal{M}, s, \mathcal{H}$ );
7   else
8     for each node  $N_c \in \{\text{children of } N\}$  do
9       | Perform prefix filter with Algorithm 3 on  $N$  to find all the
10      | candidates;
10      | GreedySearch( $N_c, \mathcal{M}, s$ );
11 end

```

---

As there is no previous study on the top- $k$  Region Aware Similarity Search problem, we extend two most relevant state-of-the-art methods to serve as the baseline: RT-Tree [10] and MHR-Tree [20]. We obtain the code of RT-Tree and MHR-Tree from the authors and make the extension based on the original code. As RT-Tree only supports keywords search, we extend it as follows: we generate prefix  $q$ -grams for each string and regard them as keywords to construct the index. When searching for a given query, we first generate its  $q$ -grams as well. Then we visit the corresponding inverted lists from top to bottom of the tree. For MHR-Tree, we first randomly verify  $k$  objects in the region and put them in the result set  $\mathcal{H}$ . Then we enumerate the threshold  $\tau$  incrementally from 0 to search for candidates until  $\tau \leq \text{UB}^{\mathcal{H}}$ .

## 6.2 Results

To evaluate the effect of proposed techniques, we implement three methods: **Simple**, **Estimation** and **Greedy**. **Simple** uses the straightforward spatial first strategy; **Estimation** is the estimation based selection strategy; and **Greedy** is the Greedy Search method that only perform linear scan on leaf nodes. The metric for evaluation is the number of candidates and query time.

Firstly, we evaluate the candidate number of each method to judge the filtering power. The result is shown in Fig. 4. It is clear **Estimation** and **Greedy** can prune more candidates than **Simple** because they can avoid visit redundant inverted lists. As **Estimation** can find the optimal combination of nodes to verify, the top- $k$  algorithm could be convergent quickly without much linear scan. Thus it involves the least number of candidates. **Greedy** also shows a strong filter power because it takes full advantage of the filter power from different levels.

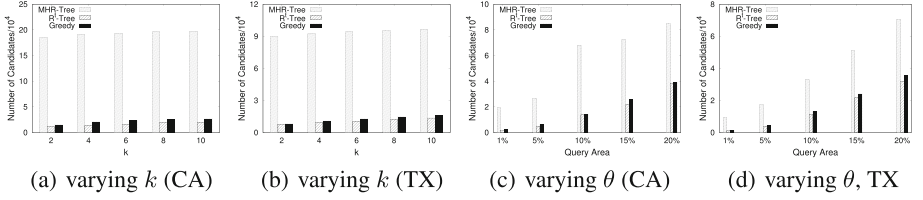


Fig. 4. Effect of proposed techniques: number of candidates

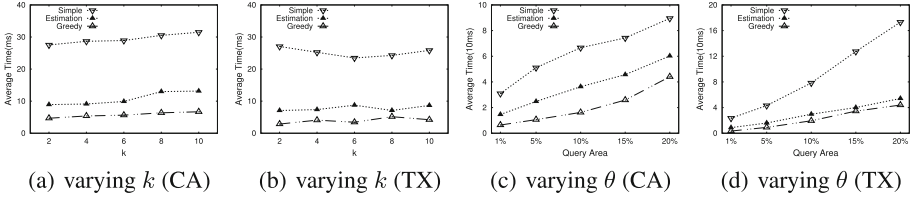


Fig. 5. Effect of proposed techniques: query time

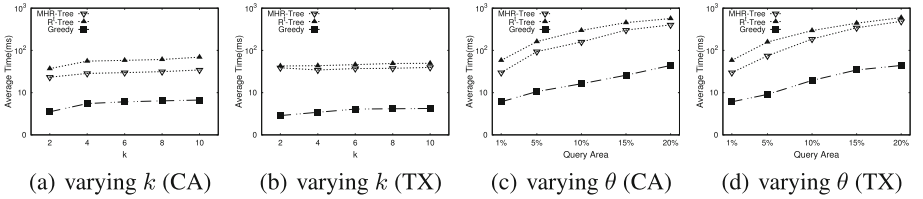


Fig. 6. Comparison with state-of-the-art methods

We then evaluate the average search time. As shown in Fig. 5, the average search times of Estimation and Greedy are much better than that of Simple because they can reduce the candidate number by performing further filtering on lower tree levels. Although Estimation shows better filter power, but it also involves relatively heavy estimating cost, so Estimation does not perform better than Greedy.

We compare our Greedy algorithm with the extended state-of-the-art methods RT-Tree- $k$  and MHR-Tree- $k$ . We evaluate the performance on the same two datasets. For each experiment, we randomly select 10,000 queries from the dataset and report the average search time. The results are shown in Fig. 6. We first fix the region size and varied the value of  $k$ , and have the following observations. MHR-Tree- $k$  outperforms RT-Tree- $k$  a little except on CA dataset, varying  $\theta$ . This is because MHR-Tree- $k$  is an approximate search algorithm, which trades accuracy for improving query performance. Our method outperforms the other two methods by one or two orders of magnitude because we propose specified filter techniques for edit distance and utilize the hierarchical index to avoid

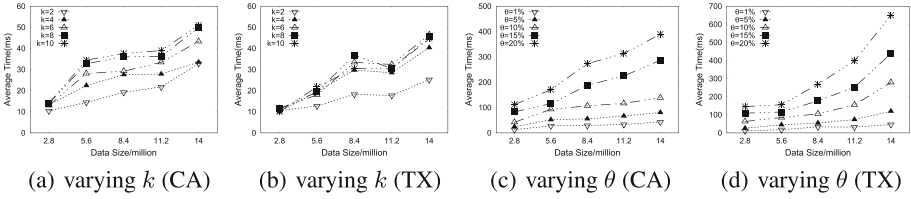


Fig. 7. Scalability

large numbers of verifications in top- $k$  search. Although the extended RT-Tree- $k$  method can also take advantage of prefix filtering, it fails to make full use of the filter power since it uses only one gram length. Thus there will be a large number of linear scan when the threshold increases beyond the filter ability. We then fix the value of  $k$  and vary the region size from 1% to 20% of the whole space. We find similar trend with the above experiments. The reason is that as we adopt spatial first strategy, it will not involve many verifications on upper levels. And we can prune a significant number of dissimilar strings with the region constraint.

Table 2. Index

Dataset	Method	Size (MB)	Construction time (s)
CA	Greedy	189	17.8
	RT-Tree- $k$	75	18.1
	MHR-Tree- $k$	1968	884
TX	Greedy	164	16.2
	RT-Tree- $k$	70	19.6
	MHR-Tree- $k$	1878	859

Table 2 shows the index size and index time of each algorithm. We can see that Greedy involves least index construction time among all the methods. Moreover, the index size of our method is much smaller than that of MHR-Tree- $k$  and comparable to RT-Tree- $k$ .

We evaluate the scalability of our algorithm. We vary the size of each datasets and test the average query time for our Greedy algorithm. As shown in Fig. 7, our method scales very well with different  $k$  values and region sizes and can support large-scale data. Our method also scales very well with different  $\theta$  values. The average search time of our method increased almost linearly with the increase of the dataset size.

## 7 Conclusion

In this paper, we solve the problem of region-aware error-tolerate search. We propose a hierarchical index for the geo-textual objects. Based on the index structure, we design a basic top- $k$  algorithm and make optimization with the idea of incremental prefix. We further improve the filter power by leveraging estimation based methods and greedy approximation. The experimental results on real world POI datasets show that our method outperforms state-of-the-art methods by 1–2 orders of magnitude.

**Acknowledgement.** This work was supported by NSFC (91646202), National Key R&D Program of China (SQ2018YFB140235), and the 1000-Talent program.

## References

1. Broder, A.Z., Charikar, M., Frieze, A.M., Mitzenmacher, M.: Min-wise independent permutations (extended abstract). In: STOC, pp. 327–336 (1998)
2. Chaudhuri, S., Ganti, V., Kaushik, R.: A primitive operator for similarity joins in data cleaning. In: ICDE, p. 5 (2006)
3. Chen, L., Cong, G., Cao, X.: An efficient query indexing mechanism for filtering geo-textual data. In: SIGMOD, pp. 749–760 (2013)
4. Chen, L., Cong, G., Jensen, C.S., Wu, D.: Spatial keyword query processing: an experimental evaluation. PVLDB **6**(3), 217–228 (2013)
5. Cong, G., Jensen, C.S.: Querying geo-textual data: spatial keyword queries and beyond. In: SIGMOD, pp. 2207–2212 (2016)
6. Cong, G., Jensen, C.S., Wu, D.: Efficient retrieval of the top-k most relevant spatial web objects. PVLDB **2**(1), 337–348 (2009)
7. Felipe, I.D., Hristidis, V., Rishe, N.: Keyword search on spatial databases. In: ICDE, pp. 656–665 (2008)
8. Gravano, L., Ipeirotis, P.G., Jagadish, H.V., Koudas, N., Muthukrishnan, S., Srivastava, D.: Approximate string joins in a database (almost) for free. In: VLDB, pp. 491–500 (2001)
9. Li, C., Lu, J., Lu, Y.: Efficient merging and filtering algorithms for approximate string searches. In: ICDE, pp. 257–266 (2008)
10. Li, G., Wang, Y., Wang, T., Feng, J.: Location-aware publish/subscribe. In: KDD, pp. 802–810 (2013)
11. Li, Z., Lee, K.C.K., Zheng, B., Lee, W., Lee, D.L., Wang, X.: IR-tree: an efficient index for geographic document search. IEEE Trans. Knowl. Data Eng. **23**(4), 585–599 (2011)
12. Mazeika, A., Böhlen, M.H., Koudas, N., Srivastava, D.: Estimating the selectivity of approximate string queries. ACM Trans. Database Syst. **32**(2), 12 (2007)
13. Navarro, G.: A guided tour to approximate string matching. ACM Comput. Surv. **33**(1), 31–88 (2001)
14. Rocha-Junior, J.B., Gkorgkas, O., Jonassen, S., Nørnvåg, K.: Efficient processing of top-k spatial keyword queries. In: SSTD, pp. 205–222 (2011)
15. Wang, J., Li, G., Deng, D., Zhang, Y., Feng, J.: Two birds with one stone: an efficient hierarchical framework for top-k and threshold-based string similarity search. In: ICDE, pp. 519–530 (2015)

16. Wang, J., Lin, C., Li, M., Zaniolo, C.: An efficient sliding window approach for approximate entity extraction with synonyms. In: EDBT (2019)
17. Wang, X., Ding, X., Tung, A.K.H., Zhang, Z.: Efficient and effective KNN sequence search with approximate n-grams. PVLDB **7**(1), 1–12 (2013)
18. Wu, J., Zhang, Y., Wang, J., Lin, C., Fu, Y., Xing, C.: A scalable framework for metric similarity join using mapreduce. In: ICDE (2019)
19. Yang, Z., Yu, J., Kitsuregawa, M.: Fast algorithms for top-k approximate string matching. In: AAAI (2010)
20. Yao, B., Li, F., Hadjieleftheriou, M., Hou, K.: Approximate string search in spatial databases. In: ICDE, pp. 545–556 (2010)
21. Zhang, C., Zhang, Y., Zhang, W., Lin, X.: Inverted linear quadtree: efficient top K spatial keyword search. In: ICDE, pp. 901–912 (2013)
22. Zhang, D., Tan, K.L., Tung, A.K.H.: Scalable top-k spatial keyword search. In: EDBT, pp. 359–370 (2013)
23. Zhang, Y., Li, X., Wang, J., Zhang, Y., Xing, C., Yuan, X.: An efficient framework for exact set similarity search using tree structure indexes. In: ICDE, pp. 759–770 (2017)
24. Zhang, Y., Wu, J., Wang, J., Xing, C.: A transformation-based framework for KNN set similarity search. IEEE Trans. Knowl. Data Eng. (2019)



# Collective POI Querying Based on Multiple Keywords and User Preference

Dongjin Yu<sup>1</sup>(✉), Yiyu Wu<sup>1</sup>, Chengfei Liu<sup>2</sup>, and Xiaoxiao Sun<sup>1</sup>

<sup>1</sup> School of Computer Science and Technology,  
Hangzhou Dianzi University, Hangzhou, China

{yudj,sunxiaoxiao}@hdu.edu.cn, wyygoup@gmail.com

<sup>2</sup> Swinburne University of Technology, Melbourne, Australia  
cliu@swin.edu.au

**Abstract.** Point-of-interest (POI) querying, which searches and recommends visiting places for individuals with context constraints, has recently become a very popular location-based service. The existing approaches, however, mainly focus on finding a single POI instead of a group of POIs that are neighbouring with each other. Some few approaches do handle the querying of collective POIs, but fail to consider users' preference. In this paper, we devise a novel approach which aims to retrieve collective POIs based on multiple keywords given by a user as well as user preference, POI popularity and congestion. In addition, we design a cost function to calculate the visit cost of the candidate POIs. We also propose an efficient algorithm based on IR-tree which finds the optimal solution to achieve the balance between multiple optimization targets. The extensive experiments based on the real data from Toronto Canada demonstrate the effectiveness and efficiency of our approach.

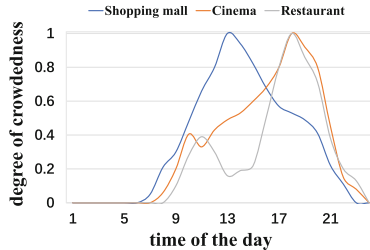
**Keywords:** Spatial keyword query · User preference · Congestion avoidance · Popularity · Collective point-of-interest · Location-based service

## 1 Introduction

During the past decade, location-based service (LBS), such as Yelp and FourSquare, has witnessed an unprecedented development with the continuous progress of location technology, as well as the popularity of mobile phones. Meanwhile, the spatial keyword query, which retrieves a set of spatial spaces that satisfy specific constraints, has attracted widespread attention both in academia and industry recently. In [1, 3, 9], a Collective Spatial Keywords Querying (CSKQ) problem is proposed, which can be illustrated as follows. Given a set of POIs  $P = \{p_1, p_2, \dots, p_n\}$  and a query  $q$ , CSKQ is used to retrieve a set of adjacent POIs that not only are close to the query location but also contain all the query keywords. Different from the existing CSKQ query research, this paper considers

two new issues: (1) When querying a set of POIs that contain all query keywords, the traditional CSKQ research usually considers only text matching between spatial POIs and query keywords, but ignores the user’s preference for these POIs. For example, when Lisa wants to find a POI with the keyword “Chinese restaurant”, and there are a couple of candidate POIs around her. Obviously she prefers the POI with a higher preference score. (2) The traditional CSKQ research only retrieves the related POIs which are close to the query location, but does not consider the optimal time required to visit the POIs. In fact, popular POIs are usually overcrowded during peak hours, resulting in long queues and poor service, which reduces user interest.

*Example 1.* Figure 1 shows three candidate POIs, and the degree of crowdedness of these POIs during a whole day, with a larger ordinate value indicating that the POI is more crowded at that time. The shopping mall, cinema and restaurant suffer the peak flow at around 13:00, 18:00, and 18:00 respectively. Assuming Mark departs from a location close to these POIs at 12:00 and wants to visit these three POIs. He enjoys lunch and sightseeing at the shopping mall between 12:00 and 15:00, watches the recently released movie at the cinema between 15:00 and 18:00, and enjoys a Mexican meal at the restaurant from 18:00 to 20:00. However, this visit sequence suffers a severe congestion according to Fig. 1. In contrast, a better choice could be: having lunch at the restaurant between 12:00 and 14:00, watching movie at the cinema between 14:00 and 17:00, having dinner and going shopping at the shopping mall between 17:00 and 20:00. As we can see, adjusting the visit order appropriately avoids visiting POIs during the crowded periods, resulting in shorter waiting time and better services during the off-peak hours.



**Fig. 1.** The degree of congestion of three POIs.

In order to solve the above problems, we propose a new personalized query called Collective Keywords Preference Query (CKPQ). CKPQ finds a group of POIs that satisfy the following conditions: (1) The size of the group does not exceed the number of the query keywords. (2) The group of POIs contains all query keywords. (3) The POIs are close to the query location. (4) The POIs in the group are close to each other. In other words, the farthest distance between any two POIs should be as close as possible. (5) The POIs satisfy the user’s preference. (6) The group of POIs is combined with a visiting order, which can avoid visiting popular POI during peak hours.

In this paper, we firstly establish a user preference model based on historical visit records of the user and the potential topics of POIs. Then we construct a cost function to measure the visit cost of the candidate group according to the distance within the group, the distance to the query location, the user's preference for the POIs and the accessibility of the POIs. Based on the cost function, we propose an algorithm based on IR-tree, which can efficiently find a group of POIs to answer the CKPQ problem. The contributions of this paper are summarized as follows:

- We propose a novel approach to the personalized collective spatial keyword query based on user preference, named Collective Keywords Preference Query (CKPQ), and design a cost function to measure the visit cost of the candidate result of CKPQ.
- We propose an efficient retrieve algorithm based on IR-tree, which can find a solution of CKPQ to achieve the balance between multiple optimization targets.
- We conduct extensive experiments based on the real data from Toronto Canada to evaluate the efficiency and effectiveness of our approach.

The rest of the paper is organized as follows. Section 2 describes the related definitions and the problem statement. Section 3 introduces the proposed approach in detail. The experimental results based on real data are presented and analyzed in Sect. 4. In Sect. 5 we show the related work, while in Sect. 6 we conclude the paper.

## 2 User Preference Model and Problem Definition

To achieve an adaptive personalized search, we present a user preference model in Sect. 2.1 to obtain a user's preference topic distribution which is composed of a set of feature words representing the preference and corresponding scores. Afterwards, in Sect. 2.2, we state some definitions and formalize the problem of CKPQ.

### 2.1 User Preference Model

The user's preference is constantly changing over time. Intuitively, we should be concerned with a user's recent visit behaviour, which represents the user's current preference. As time goes on, if the preference is not strengthened again, it will gradually be weakened. Therefore, we define a monotonically decreasing function  $f(t)$  with time  $t$  to assign the weights of visit behaviour in the user preference model. In other words, although all historical visit behaviours of a user will have an impact on the user's final preference model, the closer the visiting time is, the greater its impact would be.

We use a damping function to achieve the goal of monotonically decreasing time function  $f(t)$  given in Eq. (1):

$$f(t) = \left(1 + \frac{t}{T_0}\right)e^{-\frac{t}{T_0}} \quad (1)$$



We define a user preference vector  $PV_u$  which consists of the topic labels and their corresponding weights to represent the preference model of user  $u$ . Each visit behaviour of the user can be represented by a vector  $PV_{t_k}$ , which contains the POI potential topic set shown in Definition 1 and the user’s explicit rating of the visit behaviour. The user preference model is as follows:

$$PV_u = f(t_0)PV_{t_0} + f(t_1)PV_{t_1} + \dots + f(t_k)PV_{t_k} \tag{2}$$

where  $f(t_k)$  is the time weight at time  $t_k$  and  $PV_{t_k}$  is the preference vector of the visit POI at time  $t_k$ . After obtaining the user’s preference model  $PV_u$ , we save  $PV_u$  as the user preference topic set and the corresponding preference topic score set shown in Definition 2.

**Definition 1** (*POI Potential Topic Set*). A POI potential topic set consists of a set of words that represent the potential topics of a certain POI. Given a POI  $p$ , its potential topic set is represented as  $TS_p = \{ts_1, ts_2, \dots, ts_j, \dots, ts_s\}$ , where  $ts_j$  represents the  $j^{th}$  potential topic label where  $1 \leq j \leq s$ .

**Definition 2** (*User Preference Topic Set and User Preference Topic Score Set*). A user preference topic set contains of a set of words that represent individual’s potential topics of preference. Given a user  $u$ , its preference topic set is represented as  $TS_u = \{ts_1, ts_2, \dots, ts_j, \dots, ts_k\}$ , where  $ts_j$  represents the  $j^{th}$  preference topic label ( $1 \leq j \leq k$ ). Correspondingly, the user preference topic score set is denoted as  $UPS_u = \{UPS_{u,ts_1}, UPS_{u,ts_2}, \dots, UPS_{u,ts_j}, \dots, UPS_{u,ts_k}\}$ , where  $UPS_{u,ts_j}$  represents the score of user  $u$  on the preference topic label  $ts_j$  ( $1 \leq j \leq k$ ).

**Definition 3** (*POI Preference Score*). A POI preference score refers to the score of user  $u$  on POI  $p$ , which is calculated by the POI potential topic set  $TS_p$ , the user preference topic set  $TS_u$  and the user preference topic score set  $UPS_u$ , given as:

$$Pre(u, p) = \sum_{ts_j \in TS_u \cap TS_p} UPS_{u,ts_j} \times sc(p) \tag{3}$$

Here,  $sc(p)$  is the average rating of POI  $p$  by all visitors,  $ts_j$  is the co-occurring topic label in both the user potential topic set  $TS_u$  and the POI potential topic set  $TS_p$ .

*Example 2.* Suppose there are two restaurants with the average rating  $sc(p_1) = 3.5$  and  $sc(p_2) = 4.0$  respectively. Their POI potential topic sets are  $TS_{p_1} = \{chinese\ food, wifi-free, ambience-upscale\}$  and  $TS_{p_2} = \{fast\ food, wifi-no, ambience-casual\}$  respectively. Suppose there is a user  $u$ , whose preference topic set is  $TS_u = \{chinese\ food, wifi-free, ambience-casual\}$  with the corresponding preference topic score set  $UPS_u = \{0.4, 0.5, 0.6\}$ . Then  $Pre(u, p_1) = 3.15$  and  $Pre(u, p_2) = 2.40$ .

## 2.2 Problem Definition

**Definition 4** (*Collective Keywords Preference Query (CKPQ)*). Given a user  $q.u$  with a query location  $q.\lambda$ , a query time  $q.\tau$  and a set of keywords  $q.\psi$ , the CKPQ can be represented by  $q = (u, \lambda, \tau, \psi)$ , indicating a query for a set of POIs that have the minimum cost.

**Definition 5** (*Maximum Query Euclidean Distance*). Given a CKPQ query  $q$  and a set of POIs  $\chi$ , the maximum Euclidean distance corresponds to the maximum distance among the Euclidean distances between  $q.\lambda$  and any POI in  $\chi$ , as well as the Euclidean distance between any two POIs in  $\chi$ , given as:

$$Dist_s(q.\lambda, \chi) = \gamma * \max_{j \in \chi} Dist(q.\lambda, j) + (1 - \gamma) * \max_{i, j \in \chi} Dist(i, j) \quad (4)$$

**Definition 6** (*POI Popularity*). The POI popularity shows how much a POI is popular, indicated by the count of visits of all visitors, given as:

$$Popularity(p) = \frac{count(p)}{\max(count(i), i \in P)} \quad (5)$$

Here,  $count(p)$  is the visiting count of all users of POI  $p$  and  $P$  is the set of all POIs. For example,  $P = \{p_1, p_2, p_3\}$ ,  $count(p_1) = 50$ ,  $count(p_2) = 30$ ,  $count(p_3) = 20$ , then  $Popularity(p_1) = 1$ ,  $Popularity(p_2) = 0.6$ ,  $Popularity(p_3) = 0.4$ .

**Definition 7** (*POI Congestion*). We obtain the POI congestion of a POI by counting the occurrence  $Ocr(p, t)$  per time period for each POI of all users, given as:

$$Congestion(p, t) = \log\left(\frac{count(p)}{Ocr(p, t)}\right) \quad (6)$$

Here,  $Ocr(p, t)$  is the visit count of POI  $p$  at time period  $t$ . For example, there is a POI with the visit count  $\{100, 200, 300, 400\}$  at 9:00, 10:00, 11:00 and 12:00. Thus, totally there are 1000 visits to this POI and the maximum visit count for this POI reaches 400 at 12:00. Then the congestion during these four hours is  $\{2.30, 1.61, 1.20, 0.92\}$ . The smaller the value of congestion, so the least suitable visit time is 12:00, while the most suitable visit time is 9:00. Consequently, it is suggested to visit it at 9:00.

**Definition 8** (*POI Accessibility*). We obtain accessibility of POI  $p$  at time  $t$  based on popularity and congestion of the POI according to Eq. (7):

$$Access(p, t) = Popularity(p) \times Congestion(p, t) \quad (7)$$

**Definition 9** (*Preference Distance*). Given a set of POIs  $\chi$ , the preference distance is used to represent the accessibility and the user's overall preference of POIs  $\chi$ , given as:

$$Dist_p(q, \chi) = \frac{|\chi|}{\sum_{j \in \chi} (\beta * Access(j, q.\tau) + (1 - \beta) * Pre(q.u, j))} \quad (8)$$

Here,  $\beta \in [0, 1]$  is a weight parameter to balance accessibility and user preference.

**Definition 10** (*Cost Function*). Given a set of POIs  $\chi$ , the cost function involves two parts: maximum query Euclidean distance and preference distance, calculated as:

$$Cost(q, \chi) = \alpha * Dist_s(q, \chi) + (1 - \alpha) * Dist_p(q, \chi) \quad (9)$$

Here,  $\alpha \in [0, 1]$  is a weight parameter to balance the maximum Euclidean distance and the preference distance. The larger the value of  $\alpha$ , the higher the weight of the maximum Euclidean distance. When  $\alpha$  equals to 1, the CKPQ problem becomes a CSKQ problem.

**Problem Statement:** Given a set of POIs  $P = \{p_1, p_2, \dots, p_n\}$  and a Collective Keywords Preference Query (CKPQ)  $q = (u, \lambda, \tau, \psi)$ , our problem is to find a group of POIs  $\chi$ , where  $\chi \subseteq P$  and  $\chi$  contains all keywords in  $q.\psi$ , such that  $\chi$  is  $argmin_{\chi'} \{cost(q, \chi') \mid \chi' \subseteq P\}$ .

### 3 Approaches

For spatial keyword query, building spatial index is an effective way to solve a high-dimensional space search problem. In this section, we first introduce the IR-tree index in Sect. 3.1, we can easily change the scope of the query through IR-tree. Based on IR-tree, we then present a baseline algorithm to answer CKPQ in Sect. 3.2. Finally, we devise several pruning strategies to improve the query efficiency of the baseline algorithm and give the improved algorithm based on the pruning strategies in Sect. 3.3.

#### 3.1 IR-Tree

The IR-tree is essentially an R-tree, with a corresponding inverted file attached to each of its nodes. The inverted file stores the keyword information of the POIs in the node, which mainly includes two parts: a keyword directory and a POI list. The former stores all the keywords contained in the node, whereas the latter stores the POIs contained in each keyword. Each leaf node in the IR-tree contains an POI represented by  $(p, p.\lambda, p.di)$ , where  $p$  represents the POI,  $p.\lambda$  represents the location of POI  $p$ , and  $p.di$  represents the keyword information of POI  $p$ . Each non-leaf node in the IR-tree contains some nodes represented by  $(cp, rect, cp.di)$ , where  $cp$  represents a pointer to the child node of the non-leaf node,  $rect$  represents the minimum boundary rectangle (MBR) of the child node, and  $cp.di$  is the pseudo-text description that is a union of all keyword information in the entries of the child nodes. Figure 2(a) shows eight spatial POIs and its keywords, and Fig. 2(b) illustrates the IR-tree index structure corresponding to Fig. 2(a).

Given a CKPQ query  $q$ , we adopt an IR-tree to find the candidate solution within the query region. We initialize a min-priority queue to maintain the intermediate results and add the root of IR-tree to the queue. The node is sequentially

taken out from the queue when querying, and the child nodes including the query keywords are added to the priority queue according to the inverted file of the node, the priority value of min-priority queue is minimum distance from query location to the node. When the priority value of the fetched node from priority queue exceeds the query region, the query stops.

### 3.2 Baseline Approach

In this section, we propose a baseline approach, called Enum-Search, which enumerates all groups of POIs covering the query keywords within query region, calculates the cost of each group, and selects the group with the lowest cost as the query result.

Enum-Search finds all subsets of the POIs which contains all keywords from the query region where the center is  $q$ ,  $\lambda$  and the radius is  $r$ . If needed, we can expand the search radius  $r = r + \Delta r$  and search the result again within the new search radius.

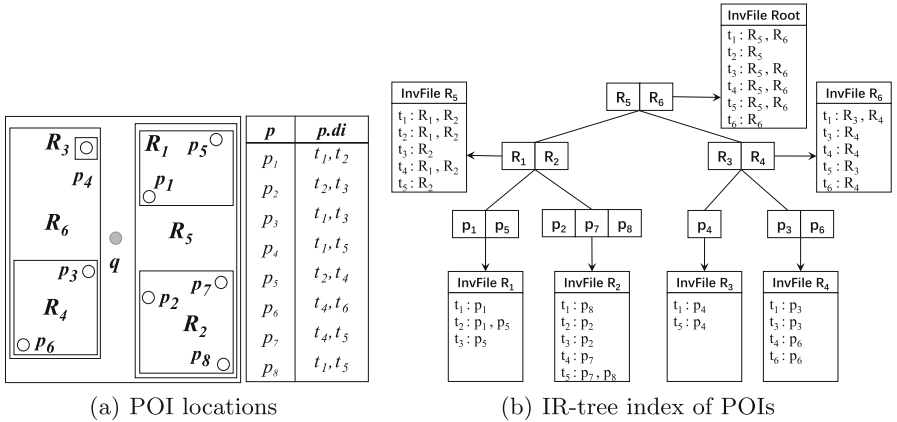


Fig. 2. An IR-tree example.

---

#### Algorithm 1. findBestOrder

---

**Input:** a candidate group  $G$  containing all query keywords, visit time  $t$

**Output:** The best visit order  $O$  of  $G$

---

- 1:  $O \leftarrow \emptyset$ ;  $congestCost \leftarrow 0$ ;
  - 2: **for** each permutation  $\chi$  of  $G$  **do**
  - 3:      $feasibleCost \leftarrow 0$ ;  $index \leftarrow 0$ ;
  - 4:     **for** each POI  $j$  in  $\chi$  **do**
  - 5:          $feasibleCost \leftarrow feasibleCost + congestion(j, t + index * \Delta t)$ ;
  - 6:          $index \leftarrow index + 1$ ;
  - 7:     **if**  $feasibleCost > congestCost$  **then**
  - 8:          $O \leftarrow \chi$ ;  $congestCost \leftarrow feasibleCost$ ;
  - 9: **return**  $O$ ;
-

Given a set of candidate POIs that contain all the query keywords, since all permutations have the same maximum query Euclidean distance, popularity, and user preferences, we only need to calculate the POI congestion of each permutation to obtain the best visit order according Definition 7. The pseudo code is shown in Algorithm 1. We use  $\Delta t$  to indicate the time difference between visiting two adjacent POIs (Line 5).

Algorithm 2 shows the pseudo code of Enum-Search. We use a POI set  $C = \{C_1, C_2, \dots, C_z\}$ , where  $C_z$  represents the set to save all POIs containing keyword  $z$  (Line 6). In addition, we use  $\text{minDist}(q, e)$  to calculate the minimum distance between node  $e$  in IR-tree and the query location,  $\text{minDist}(q, e)$  represents the distance between POI  $e$  and query location when  $e$  is a leaf node, while  $e$  is a non-leaf node,  $\text{minDist}(q, e)$  represents the shortest distance between the *rect* of node  $e$  and query location. If  $\text{minDist}(q, e) > r$ , node  $e$  will not be considered during retrieval (Line 9).

---

### Algorithm 2. Enum-Search

---

**Input:** a CKPQ query  $q$ , an *irTree* containing all POIs

**Output:** a query result  $R$  and its corresponding result cost  $RC$

---

```

1:  $U \leftarrow \text{new queue}; U.\text{add}(\text{irTree.root}, 0)$ ;
2:  $\text{feasibleCost} \leftarrow 0; C \leftarrow \emptyset$ ;
3: while  $U$  is not empty do
4:    $e \leftarrow U.\text{poll}()$ ;
5:   if  $e$  is an object and  $e.\psi \cap q.\psi \neq \emptyset$  then
6:      $C_{t_i} \leftarrow C_{t_i} \cup e, t_i = e.\psi \cap q.\psi$ ;
7:   else if  $e$  is a non-leaf node then
8:     for each  $e'$  in node  $e$  do
9:       if  $e' \cap q.\psi \neq \emptyset$  and  $\text{minDist}(q, e') < r$  then  $U.\text{add}(e')$ ;
10:  $\text{groupList} \leftarrow \{p_1, p_2, \dots, p_z \mid p_1 \in C_{t_1}, p_2 \in C_{t_2}, \dots, p_z \in C_{t_z}\}$ ;
11:  $R \leftarrow \emptyset; RC \leftarrow +\infty$ ;
12: for each group in  $\text{groupList}$  do
13:    $\chi \leftarrow \text{findBestOrder}(\text{group}, q, \tau)$ ;
14:    $\text{feasibleCost} \leftarrow \text{Cost}(q, \chi)$ ;
15:   if  $\text{feasibleCost} < RC$  then
16:      $RC \leftarrow \text{feasibleCost}; R \leftarrow \chi$ ;
17: return  $R$  and  $RC$ ;

```

---

Since the enumeration approach needs to query all POIs that contain the query keywords in the space, and randomly combines these POIs and calculates their visit costs, its poor query efficiency makes it difficult for users to accept.

### 3.3 A Nearest-Distance-First Approach with Pruning

Although the query result and visit cost obtained by Enum-Search is optimal, its running time is difficult to be accepted especially when searching among a large group of POIs. Therefore, we employ a pruning based algorithm, called RKD-Search, which considers the rare keyword and distances. We first employ a simple nearest distance search algorithm called ND-Search to find a feasible

result as the current best result, then use two kinds of pruning strategies to improve the query efficiency of the search algorithm.

ND-Search finds the POIs closest to the query location  $q.\lambda$  for each keyword  $t_i$  in  $q.\psi$  from the query region, then combines these POIs to constitute the query results. The pseudo code is shown in Algorithm 3.

We use *keySet* to save the keywords that have not yet been queried. At each iteration, we take the head element  $e$  from  $U$ , if  $e$  is a leaf node, add it to the result set, and remove the keyword contained in the *keySet* (Lines 5–7). If  $e$  is a non-leaf node in the IR-tree, judge whether the node contains the keyword contained in *keySet*, and inserts all child nodes which contain the keyword contained in *keySet* into  $U$  (Lines 9–11).

---

### Algorithm 3. ND-Search

---

**Input:** a CKPQ query  $q$ , an *irTree* containing all POIs

**Output:** a query result  $R$  and its corresponding result cost  $RC$

---

```

1:  $U \leftarrow$  new priority queue;  $U.add(irTree.root, 0)$ 
2:  $R \leftarrow \emptyset$ ;  $RC \leftarrow 0$ ;  $keySet \leftarrow q.\psi$ ;
3: while  $U$  is not empty do
4:    $e \leftarrow U.poll()$ ;
5:   if  $e$  is an object and  $e.\psi \cap q.\psi \neq \emptyset$  then
6:      $R.add(e)$ ;
7:      $keySet.remove(e.\psi)$ ;
8:   else if  $e$  is a non-leaf node then
9:     for each  $e'$  in node  $e$  do
10:      if  $e'.\psi \cap q.\psi \neq \emptyset$  and  $minDist(q, e') < r$  then
11:         $U.add(e', minDist(q, e'))$ ;
12:   if  $keySet = \emptyset$  then break;
13:  $RC \leftarrow Cost(q, R)$ ;
14: return  $R$  and  $RC$ 

```

---

After we have obtained the current best visit cost, we use two pruning strategies to improve the query efficiency of the algorithm. Two pruning strategies are as follows:

**Lemma 1** (*Pruning Strategy 1*). *Given a CKPQ query  $q$  and the current best visit cost  $curCost$ , any POI or node whose distance to query location  $q.\lambda$  exceeds  $curCost/(\alpha * \gamma)$  can be pruned, where  $\alpha$  is the weight parameter in Definition 10 and  $\gamma$  is the weight parameter in Definition 5.*

*Proof.* (1) Assuming node  $e$  is a POI containing a query keyword, and its distance to  $q.\lambda$  is  $Dist(q.\lambda, e)$ , if  $Dist(q.\lambda, e) > curCost/(\alpha * \gamma)$ , according to Definition 5:  $\max_{j \in \chi} Dist(q.\lambda, j) \geq Dist(q.\lambda, e) > curCost/(\alpha * \gamma)$ , we obtain that  $Dist_s(q, \chi) = \gamma * \max_{j \in \chi} Dist(q.\lambda, j) + (1 - \gamma) * \max_{i, j \in \chi} Dist(i, j) > curCost/\alpha$ . Further, according to Definition 10, we obtain:  $Cost(q, \chi) = \alpha * Dist_s(q, \chi) + (1 - \alpha) * Dist_p(q, \chi) > curCost + (1 - \alpha) * Dist_p(q, \chi)$ . Therefore, the candidate group of POIs containing POI  $e$  must be higher than the current best visit cost  $curCost$ . In other words,  $e$  can be pruned.

(2) Assuming node  $e$  is a non-leaf with a query keyword, with the shortest distance of  $\min Dist(q, e) = curCost/(\alpha*\gamma)$  between all POIs in  $e$  and the query location  $q.\lambda$ , if  $e$  contains a candidate POI, then the distance between the POI and the query location must exceed  $curCost/(\alpha*\gamma)$ . Then we use the same proof method in PROOF (1) when node  $e$  is a POI containing a query keyword.

Secondly, since the query result of the CKPQ problem must contain all the keywords provided, we choose to prioritize the rare keyword when querying. For each keyword, we pre-calculate the number of POIs that contain this keyword, and sort the query keywords in the ascending order of the number of containing POIs. The rare keyword is the keyword with the least number of containing POIs in query keywords.

Considering that a query result must contain all keywords, and the distance between each pair of POIs in the result must be within a certain range, the other POIs must be located near the POI with the rare keyword. Therefore, we consider preferentially finding the POI with the rare keyword in spatial data. When querying, we first find the POIs with rare keyword, then we find the remaining keywords within a query region according to Lemma 2 based on the current best visit cost and distance between the query location and the POI with the rare keyword.

**Lemma 2** (*Pruning Strategy 2*). *Given a CKPQ query  $q$ , the current best visit cost  $curCost$ , and a POI  $e$  containing the rare keyword, the query circular region  $S$  with its center of  $e.\lambda$  and its radius of  $curCost/(\alpha*(1-\gamma))-\gamma*Dist(q,\lambda,e)/(1-\gamma)$ , then all nodes outside  $S$  can be pruned.*

*Proof.* Given a CKPQ query  $q$ , the current best visit cost  $curCost$  and a POI  $e$  containing the rare keyword, if there is a POI  $p$  with the distance to  $e$  exceeds  $curCost/(\alpha*(1-\gamma))-\gamma*Dist(q,\lambda,e)/(1-\gamma)$ , then according to Definition 5:  $\max_{j \in \chi} Dist(q,\lambda,j) \geq Dist(q,\lambda,e)$ ,  $\max_{i,j \in \chi} Dist(i,j) \geq Dist(e,p) \geq curCost/(\alpha*(1-\gamma))-\gamma*Dist(q,\lambda,e)/(1-\gamma)$ , and  $Dist_s(q,\chi) = \gamma * \max_{j \in \chi} Dist(q,\lambda,j) + (1-\gamma) * \max_{i,j \in \chi} Dist(i,j) \geq \gamma * Dist(q,\lambda,e) + (1-\gamma) * (curCost/(\alpha*(1-\gamma))-\gamma*Dist(q,\lambda,e)/(1-\gamma)) = curCost/\alpha$ . Further, according to Definition 10, we obtain  $Cost(q,\chi) = \alpha * Dist_s(q,\chi) + (1-\alpha) * Dist_p(q,\chi) \geq curCost$ . In other words, the visit cost of the feasible group including POI  $p$  and POI  $e$  with the rare keyword exceeds the current best cost  $curCost$ ,  $p$  can be pruned.

We use IR-tree to find the POI containing the rare keyword, and find the remaining keywords in the region  $S$  around the POIs with the rare keyword according to the above two lemmas. The pseudo code is shown in Algorithm 4.

We insert the node from IR-tree into the priority queue  $U$  in turn, and take the header element  $e$  from  $U$  at each iteration. We prune the element  $e$  with a query distance greater than  $curCost/(\alpha*\gamma)$  according to Lemma 1 (Line 6). If  $e$  is a POI, we search the remaining keywords in the region  $S$  according to Lemma 2. It is worth mentioning that all nodes whose distance to  $e.\lambda$  exceeds  $curCost/(\alpha*(1-\gamma))-\gamma*Dist(q,e)/(1-\gamma)$  will be pruned (Line 15 and Line 24).

## 4 Experiment

### 4.1 Data Preparation and Experimental Setting

**Data Preparation.** In order to evaluate the effectiveness and efficiency of our approach, we conduct extensive experiments on real datasets of Toronto collected from Yelp Dataset Challenge<sup>1</sup>, which covers 26,520 POIs between  $-80.12E-78.89E$  and  $42.98N-44.19N$ . As shown in Table 1, each POI includes attributes such as id, name, coordinate (latitude and longitude) and categories. In addition, each POI is also attached with a feature attribute. For example, the POI with *POI ID* = 00001 in Table 1 is attached with {GoodforMeal: dinner, NoiseLevel: quiet, Ambience: intimate, Price Range: 3}. For this experiment, we investigated

---

#### Algorithm 4. RKD-Search

---

**Input:** a CKPQ query  $q$ , an *irTree* containing all POIs

**Output:** a query result  $R$  and its corresponding result cost  $RC$

---

```

1:  $U \leftarrow$  new priority queue;  $U.add(irTree.root, 0)$ 
2:  $R, RC \leftarrow ND\text{-Search}(q, irTree)$ ;
3:  $Keyword_{rare} \leftarrow$  rare keyword in  $q$ ;
4: while  $U$  is not empty do
5:    $e \leftarrow U.poll()$ ;  $keywords \leftarrow q.\psi$ ;
6:   if  $minDist(q, e) \geq curCost/(\alpha * \gamma)$  or  $minDist(q, e) \geq r$  then break;
7:   if  $e$  is not an object then
8:     for each  $e'$  in node  $e$  do
9:       if  $e'.\psi \cap Keyword_{rare} \neq \emptyset$  then  $U.add(e', minDist(q, e'))$ ;
10:  else
11:     $keySet \leftarrow keywords.remove(e.\psi)$ ;  $C \leftarrow \emptyset$ ;
12:     $U_2 \leftarrow$  new priority queue;  $U_2.add(irTree.root, 0)$ ;
13:    while  $U_2$  is not empty do
14:       $v \leftarrow U_2.poll()$ ;
15:      if  $minDist(e, v) \geq (curCost/(\alpha * (1 - \gamma)) - \gamma * Dist(q, e)/(1 - \gamma))$  or  $minDist(q, v) \geq r$  then break;
16:      if  $v$  is not an object then
17:        for each  $v'$  in node  $v$  do
18:          if  $v'.\psi \cap keySet \neq \emptyset$  then  $U_2.add(v', minDist(e, v'))$ ;
19:          else  $C_{t_i} \leftarrow C_{t_i} \cup v$ ,  $t_i = v.\psi \cap q.\psi$ ;
20:     $groupList \leftarrow \{p_1, p_2, \dots, p_z \mid p_1 \in C_{t_1}, p_2 \in C_{t_2}, \dots, p_z \in C_{t_z}\}$ ;
21:     $feasibleCost \leftarrow +\infty$ ;
22:    for each group in  $groupList$  do
23:       $\chi \leftarrow findBestOrder(group, q, \tau)$ ;
24:      if  $minDist(e, \chi) < (curCost/(\alpha * (1 - \gamma)) - \gamma * Dist(q, e)/(1 - \gamma))$  then
25:         $feasibleCost \leftarrow Cost(q, \chi)$ ;
26:        if  $feasibleCost < RC$  then
27:           $RC \leftarrow feasibleCost$ ;  $R \leftarrow \chi$ ;
28: return  $R$  and  $RC$ 

```

---

<sup>1</sup> <https://www.yelp.com/dataset>.



1,638,087 real visit records of 22,389 users from July 26, 2015 to July 26, 2017, with 307,985 records in 2015, 808,740 records in 2016, and 521,362 records in 2017, respectively.

**Parameters Setting.** The parameters and their corresponding values in our experiment are summarized in Table 2, while the default values are indicated in bold.

**Table 1.** Example of POIs.

POI ID	Name	Latitude	Longitude	Categories
00001	Segovia	43.6651	-79.3856	restaurants, tapas bars
00002	Titika	43.6494	-79.3928	shopping, sports wear
00003	Camp 4	43.6494	-79.4218	nightlife, dive bars
...	...	...	...	...

**Table 2.** Parameter setting.

Parameter	Values (default in bold)
the number of keywords ( $ q.\chi $ )	2, <b>3</b> , 4, 5, 6
the query radius $r$	1, 2, <b>3</b> , 4, 5, 6, 7, 8
the weight parameter $\gamma$	<b>0.5</b>
the weight parameter $\beta$	<b>0.5</b>
the weight parameter $\alpha$	<b>0.5</b>
the half-life parameter $T_0$	<b>30</b>
the time difference $\Delta t$	<b>1</b>

**Query Generation.** In the query generation process, we generated five spatial keyword query sets according to the number of keywords, namely 2, 3, 4, 5 and 6. More specifically, we first randomly selected a POI from the dataset and used its location as the query location. Afterwards, we sorted all keywords in descending order by their frequency in the spatial POI, and randomly selected *top-k* keywords. The percentage of at least one keyword in each set of queries ranges from 40% to 70%, while the percentage of other keywords is less than 40%.

All algorithms were implemented in Java on Windows 10, and run on an Intel(R) CPU i5-6500 @3.19 GHz with 8 GB RAM.

## 4.2 Experimental Results

**Effect of the Number of Keywords.** We generated 5 types of queries with different values of  $|q.\psi|$ , i.e., with the number of query keywords of 2, 3, 4, 5 and 6 respectively. Figure 3 shows the running time of RKD and ENUM when the

number of keywords increases from 2 to 6. As it indicates, since ENUM needs to find all POIs containing the query keywords and randomly combines these POIs, the running time of ENUM increases rapidly as the number of keywords increases. RKD demonstrates the similar variation trend. However, since RKD employs some pruning strategies to reduce the scope of the query, its running time is much lower than ENUM, and the gap between the two approach widens as the number of keywords increases.

**Effect of Query Scope.** This test demonstrates the impact of query radius  $r$  to the performance, which limits the scope of the query. Table 3 illustrate the running time and the results of ENUM and RKD with 3 keywords, when  $r$  increases from 1 (Km) to 8 (Km). As it indicates, with the increase of  $r$ , the time consumed by ENUM increases dramatically. In contrast, although the time spent by RKD also becomes longer, it stops when  $r$  hits 5km. The reason is that RKD employs some pruning strategies to reduce the scope of the query. Therefore, it stops the query when no better solution is found within the scope. More importantly, RKD obtains consistent solutions with ENUM.

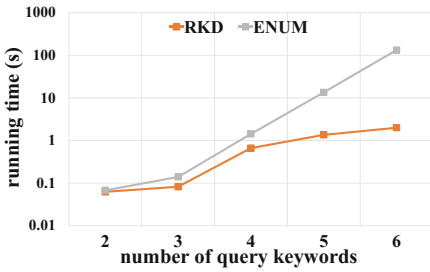


Fig. 3. Impact of parameter  $|q.\chi|$ .

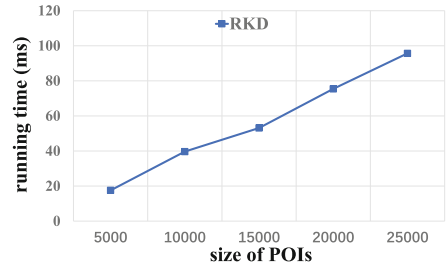


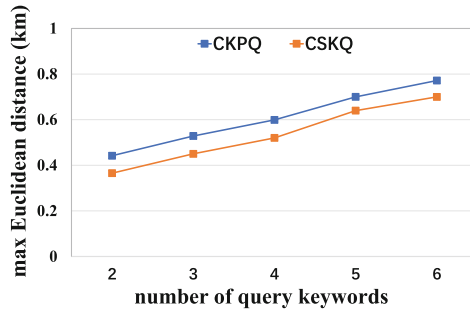
Fig. 4. Impact of the number of POIs.

Table 3. Impact of parameter  $r$  on query result.

$r$ (km)	Search algorithm			
	Enum		RKD	
	Time (ms)	Cost	Time (ms)	Cost
1	67	0.617	47	0.617
2	139	0.502	81	0.502
3	809	0.478	93	0.478
4	3630	0.472	95	0.472
5	9781	0.472	96	0.472
6	13168	0.472	96	0.472
7	15238	0.472	96	0.472
8	16847	0.472	96	0.472

**Effect of the Number of POIs in  $P$ .** Figure 4 shows the running time of RKD when the number of POIs increases from 5000 to 25000. As it indicates, as the number of the POIs increases, the running time of RKD also increases gradually. Even if the number of POIs reaches 25000, the running time keeps within a reasonable range.

**Comparison Between CSKQ and CKPQ.** In order to verify the effectiveness of CKPQ, we compare the maximum query Euclidean distances of exact results between CKPQ and CSKQ. CSKQ employs the maximum Euclidean distance cost function given in Definition 5 which is the same as CKPQ. Figure 5 illustrates the change of maximum Euclidean distance in which the number of query keywords is set to 3 and the query radius is set to 3 km for both CKPQ and CSKQ.



**Fig. 5.** Maximum query Euclidean distance of result of CKPQ and CSKQ.

It can be clearly concluded from Fig. 5 that the maximum query Euclidean distance of the exact result increases gradually with more number of query keywords for either CSKQ or CKPQ. In addition, the maximum query Euclidean distance of the POIs obtained by CKPQ is slightly larger than that of POIs obtained by CSKQ. It is because CKPQ aims to not only find a set of adjacent POIs that are close to the user, but also consider the user's preference, the popularity and congestion of these POIs.

To further verify the effectiveness of CKPQ, we handled the following experiment among 30 students. We randomly generated 10 queries, and obtained the exact results of CSKQ and CKPQ of these 10 queries. Then, we gave the scenario information of these queries such as user preference topic set, POIs potential topic set, popularity and congestion of these POIs, and let all students compare the POIs obtained by CSKQ and CKPQ. The effectiveness of the CKPQ query results is measured by the satisfaction of these 30 students. We finally obtained 300 comparison results, of which CKPQ received 294 satisfactory votes. In other words, more than 98% of students prefer the POIs obtained by CKPQ. Therefore, although the POIs obtained by CSKQ have shorter maximum query Euclidean

distance, the POIs obtained by CKPQ are more popular and more in line with the user's taste. Meanwhile, CKPQ also recommends the visit order of POIs according to the congestion of the POIs, which avoids visiting popular POIs during peak hours.

## 5 Related Work

A popular research in spatial keyword query is to find Top-k objects in space based on a ranking function which considers both spatial position and text relevance called top-k kNN query. In [5,8,11], top-k kNN query was studied in the Euclidean space, the method to handle the queries is to use IR-tree for spatial proximity querying [5,8], whereas a novel index was proposed in [11] to improve the performance of top-k spatial keyword queries named Spatial Inverted Index (S2I). Besides, [4,10,12,15] applied top-k kNN query into road network. Cho et al. proposed a novel algorithm called ALPS to minimize the number of data objects, their methodology groups objects in a road segment and converts grouped objects into a data segment [4]. In addition, Rocha-Junior et al. proposed a novel algorithm that improves query processing performance by avoiding examining the spatial neighborhood of the data objects during query execution [12]. Meanwhile, Yuan et al. presented a new problem: kNN search on road networks by incorporating social influence (RSkNN) [15], whereas Luo et al. proposed a new distributed index scheme called NPD-index to answer two types of spatial-keyword queries in a distributed setting [10].

Some other studies on spatial keywords query focus on finding a set of objects as a solution to meet the user's needs. In [16,17], Zhang et al. proposed a novel spatial keywords query called the m-closest keywords (mCK) query that aims to find the spatially closest tuples which match m user-specified keywords. Cao et al. proposed the collective spatial keyword query problem (CoSKQ) to retrieve a group of objects that the keywords contained in all objects collectively cover the query keyword [1,2]. It is worth mentioning that previous works only focus on the CoSKQ problem in the Euclidean space, Guo et al. studied the problem of collective spatial keywords query processing on road networks [6]. Besides, in [13], Sun et al. investigated the spatial keywords query with semantics which targets to find objects that is optimum regarding to both spatial proximity and semantic relevance. There are also some indexes to support efficient processing on CoSKQ and top-k kNN, such as quadtree [7], bR\*-tree [16], IR-tree [5] and AP-Tree [14].

As far as we know, most of the existing methods only consider the spatial distance in spatial keywords query, but fail to consider users' preference and the accessibility of POIs. Therefore, in this paper, we propose a new personalized collective spatial keywords query problem named Collective Keywords Preference Query (CKPQ) that can determine the group of POIs according to the distance within the group, the distance to the query location, user's potential preference and the accessibility of the POIs. Meanwhile, we propose an IR-tree based pruning algorithm to answer the CKPQ problem.

## 6 Conclusion

In this paper, we propose a novel personalized collective spatial keywords query problem named Collective Keywords Preference Query (CKPQ), and present two pruning strategies based on IR-tree to answer CKPQ. We first devise a user preference model based on visit history and the potential topic of POIs. A cost function is then constructed to calculate the cost of the candidate group based on the model. Thirdly, we develop an efficient algorithm based on IR-tree and the pruning strategies. Finally, we conduct extensive experiments based on the real data from Toronto Canada, which proves the effectiveness and efficiency of our approach.

In the future, we will improve our proposed approach in the following aspects.

(1) To speed up the query efficiency, we will consider some approximate algorithms to solve CKPQ especially when handling large-scale data sets. (2) To more intuitively demonstrate the effectiveness of CKPQ, we will conduct the comparative study on some specific query instances. (3) To enrich the solution of CKPQ, we will investigate and employ some well-established approaches to route search/recommendation problems.

## References

1. Cao, X., Cong, G., Guo, T., Jensen, C.S., Ooi, B.C.: Efficient processing of spatial group keyword queries. *ACM Trans. Database Syst.* **40**(2), 13 (2015)
2. Cao, X., Cong, G., Jensen, C.S., Ooi, B.C.: Collective spatial keyword querying. In: *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, pp. 373–384. ACM (2011)
3. Chan, H.K.H., Long, C., Wong, R.C.W.: On generalizing collective spatial keyword queries. *IEEE Trans. Knowl. Data Eng.* **30**(9), 1712–1726 (2018)
4. Cho, H.J., Kwon, S.J., Chung, T.S.: ALPS: an efficient algorithm for top-k spatial preference search in road networks. *Knowl. Inf. Syst.* **42**(3), 599–631 (2015)
5. Cong, G., Jensen, C.S., Wu, D.: Efficient retrieval of the top-k most relevant spatial web objects. *Proc. VLDB Endowment* **2**(1), 337–348 (2009)
6. Gao, Y., Zhao, J., Zheng, B., Chen, G.: Efficient collective spatial keyword query processing on road networks. *IEEE Trans. Intell. Transp. Syst.* **17**(2), 469–480 (2016)
7. Hong, H.J., Chiu, G.M., Tsai, W.Y.: A single quadtree-based algorithm for top-k spatial keyword query. *Pervasive Mob. Comput.* **42**, 93–107 (2017)
8. Li, Z., Lee, K.C., Zheng, B., Lee, W.C., Lee, D., Wang, X.: IR-tree: an efficient index for geographic document search. *IEEE Trans. Knowl. Data Eng.* **23**(4), 585–599 (2011)
9. Long, C., Wong, R.C.W., Wang, K., Fu, A.W.C.: Collective spatial keyword queries: a distance owner-driven approach. In: *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pp. 689–700. ACM (2013)
10. Luo, S., Luo, Y., Zhou, S., Cong, G., Guan, J., Yong, Z.: Distributed spatial keyword querying on road networks. In: *EDBT*, pp. 235–246. Citeseer (2014)

11. Rocha-Junior, J.B., Gkorgkas, O., Jonassen, S., Nørnvåg, K.: Efficient processing of top-k spatial keyword queries. In: Pfoser, D., Tao, Y., Mouratidis, K., Nascimento, M.A., Mokbel, M., Shekhar, S., Huang, Y. (eds.) SSTD 2011. LNCS, vol. 6849, pp. 205–222. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-22922-0\\_13](https://doi.org/10.1007/978-3-642-22922-0_13)
12. Rocha-Junior, J.B., Vlachou, A., Doulkeridis, C., Nørnvåg, K.: Efficient processing of top-k spatial preference queries. *Proc. VLDB Endowment* **4**(2), 93–104 (2010)
13. Sun, J., Xu, J., Zheng, K., Liu, C.: Interactive spatial keyword querying with semantics. In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 1727–1736. ACM (2017)
14. Wang, X., Zhang, Y., Zhang, W., Lin, X., Wang, W.: AP-tree: efficiently support continuous spatial-keyword queries over stream. In: *Proceedings of the 2015 IEEE 31st International Conference on Data Engineering (ICDE)*, pp. 1107–1118. IEEE (2015)
15. Yuan, Y., Lian, X., Chen, L., Sun, Y., Wang, G.: RSkNN: kNN search on road networks by incorporating social influence. *IEEE Trans. Knowl. Data Eng.* **28**(6), 1575–1588 (2016)
16. Zhang, D., Chee, Y.M., Mondal, A., Tung, A.K., Kitsuregawa, M.: Keyword search in spatial databases: towards searching by document. In: *Proceedings of the 2009 IEEE 25th International Conference on Data Engineering (ICDE)*, pp. 688–699. IEEE (2009)
17. Zhang, D., Ooi, B.C., Tung, A.K.: Locating mapped resources in web 2.0. In: *Proceedings of the 2010 IEEE 26th International Conference on Data Engineering (ICDE)*, pp. 521–532. IEEE (2010)



# DPSCAN: Structural Graph Clustering Based on Density Peaks

Changfa Wu, Yu Gu<sup>(✉)</sup>, and Ge Yu

School of Computer Science and Engineering,  
Northeastern University, Shenyang 110819, Liaoning, China  
guyu@mail.neu.edu.cn

**Abstract.** Structural graph clustering is one of the fundamental problems in managing and analyzing graph data. The structural clustering algorithm SCAN is successfully used in many applications because it obtains not only clusters but also hubs and outliers. However, the results of SCAN heavily depend on two sensitive parameters,  $\epsilon$  and  $\mu$ , which may result in loss of accuracy and efficiency. In this paper, we propose a novel Density Peak-based Structural Clustering Algorithm for Networks (DPSCAN). Specifically, DPSCAN clusters vertices based on the structural similarity and the structural dependency between vertices and their neighbors, without tuning parameters. Through theoretical analysis, we prove that DPSCAN can detect meaningful clusters, hubs and outliers. In addition, to improve the efficiency of DPSCAN, we propose a new index structure named DP-Index, so that each vertex needs to be visited only once. Finally, we conduct comprehensive experimental studies on real and synthetic graphs, which demonstrate that our new approach outperforms the state-of-the-art approaches.

## 1 Introduction

In recent years, graph clustering has emerged as an important primitive in a wide range of data analysis tasks. Many types of graph clustering methods have been proposed, including graph partitioning [18], propagation-based methods [4, 13] and modularity-based methods [3, 11]. While most of the existing methods successfully find clusters, they are generally unable to detect hubs and outliers. To distinguish the different roles of the vertices, a structural graph clustering algorithm named SCAN is proposed in [23]. The main concept of SCAN is that two vertices belong to the same cluster if they are similar enough. If a vertex does not belong to any cluster, it is a hub if its neighbors belong to more than one cluster, and an outlier otherwise. Due to the high time complexity of computing the structural similarities, SCAN is not scalable to large graphs. Many literatures [2, 10, 19] have been proposed to overcome the drawback of SCAN.

However, most of structural graph clustering algorithms depend on two sensitive parameters, namely  $\epsilon$  and  $\mu$ , and the change of input parameter values may heavily influence the clustering result [22]. To obtain relatively reasonable clusters, users need to run the algorithm many times to tune the parameters.

In this paper, we propose a novel structural graph clustering algorithm based on density peaks (DPSCAN) without parameters to be tuned. Our proposal is to identify the sets of clusters, hub vertices and outlier vertices. The basic idea of DPSCAN is that a vertex and its structure-dependent neighbor belong to the same cluster. Based on Density Peak Clustering [16], we firstly define three metrics for each vertex and generate the decision graph. Through the decision graph, we find all the density peaks and noise vertices, and then get the clusters grown from the density peaks. Finally, we identify the noise vertices as hubs or outliers. As illustrated in Table 1, compared with GS\*-Query, LPA, Modularity and GN, DPSCAN is the fastest graph clustering algorithm, which can detect not only clusters but also hubs and outliers.

Our main contributions can be summarized as follows:

1. *The first DP – based algorithm for structural graph clustering.* We propose a DP-based structural graph clustering algorithm, named DPSCAN, which can detect clusters, hubs, and outliers without tuning parameters. To the best of our knowledge, this is the first DP-based algorithm for structural graph clustering. Through theoretical analysis and experimental evaluation, we demonstrate that DPSCAN can find meaningful clusters and identify hubs and outliers.
2. *Flexible clustering.* Three new metrics, the local density, the dependent similarity and the dependent vertex, are first defined for each vertex in the graph. Based on the metrics, a new user-friendly decision graph is proposed specifically for graphs. Through the decision graph, DPSCAN can identify a variable number of clusters or automatically detect clusters according to user’s requirements.
3. *Efficiency.* We propose a new index structure named DP-Index. Based on DP-Index, DPSCAN can online-compute reasonable clusters efficiently, and the time complexity is only linear to the number of vertices. In addition, we further propose two optimization techniques for DP-Index. The experimental results demonstrate that our algorithm can achieve significant speedup compared to the state-of-the-art algorithms.

## 2 Related Work

**Density Peak Clustering.** Density Peak Clustering (DPC) [16] is a novel clustering algorithm recently proposed by Rodriguez and Laio in Science. As an effective and powerful tool for the task of clustering, DPC has been widely applied to many clustering problems. OCDDP, proposed in [1], utilizes a similarity-based method to set distances among nodes to find overlapping communities. An evidential community detection algorithm is explored in [25], which detects community centers based on [16] and assigns remaining nodes through a label propagation strategy [24]. However, due to inherently different problem definitions, these methods cannot be applied to structural graph clustering.



**Structural Graph Clustering.** To distinguish the different roles of the vertices, a structural graph clustering method named SCAN is proposed in [23]. SCAN successfully identifies clusters, hubs and outliers. Due to the high time complexity of computing the structural similarities, it is not scalable to large graphs. To improve the efficiency of SCAN, LinkSCAN\*, proposed in [10], adopts approximation techniques by sampling edges to reduce the number of similarity computations. SCAN++ [19] is proposed to avoid computing similarity between vertices that are shared between the neighbors of a vertex and its two-hop-away vertices. In addition, [2] proposes an algorithm named pSCAN. It avoids similarity computations by maintaining an upper bound and a lower bound for the number of similar neighbors of each vertex. However, these methods heavily depend on two sensitive parameters, namely  $\epsilon$  and  $\mu$ . To solve the problem of frequently parameters tuning, SHRINK, proposed in [8], combines modularity-based methods and structural similarity to compute clusters. [21] proposes the algorithm gSkeletonClu, which finds the optimal  $\epsilon$  and clusters vertices based on a tree-decomposition-based method. [22] proposes an index-based method. It provides an efficient algorithm GS\*-Query based on GS\*-Index to answer the query for any possible  $\epsilon$  and  $\mu$ , which is significantly faster than previous methods. In this paper, our work tries to extend DPC [16] and develop an efficient structural graph clustering algorithm without tuning parameters.

**Other Graph Clustering Models.** Other graph clustering methods have also been studied in the literature, which include the betweenness-based method [6], the modularity-based method [3, 11], the propagation-based method [4, 13, 14] and graph partitioning [12, 18]. [13] is a recently proposed graph clustering approach named FluidC, which outperforms most of the existing methods. It is based on the idea of fluids interacting by expanding and pushing each other. While all these graph clustering methods successfully find clusters, they do not distinguish hubs and outliers in a graph.

### 3 DP-Based Structural Graph Clustering

#### 3.1 The Notion of Structure-Dependent Clusters

In this paper, we focus on an unweighted undirected graph  $G = (V, E)$  [2], where  $V$  is the set of vertices and  $E$  is the set of edges. We denote the number of vertices  $|V|$  and the number of edges  $|E|$  by  $n$  and  $m$  respectively.

**Definition 1 (Structural Neighborhood).** *The structural neighborhood of vertex  $u$ , denoted by  $N[u]$ , is defined as  $N[u] = \{v \in V | (u, v) \in E\} \cup \{u\}$ .*

Note that, the degree of  $u$ , denoted by  $deg[u]$ , is the cardinality of  $N[u]$  (i.e.,  $deg[u] = |N[u]|$ ). The open neighborhood of  $u$ , denoted by  $N(u)$ , is the set of neighbors of  $u$  (i.e.,  $N(u) = \{v \in V | (u, v) \in E\}$ ).

**Definition 2 (Structural Similarity).** *The structural similarity between two vertices  $u$  and  $v$ , denoted by  $\sigma(u, v)$ , is defined as the number of common structural neighbors between  $u$  and  $v$ , normalized by the geometric mean of their cardinalities of the structural neighborhood. That is*

$$\sigma(u, v) = \frac{|N[u] \cap N[v]|}{\sqrt{\deg[u] \cdot \deg[v]}} \quad (1)$$

Intuitively, we can see that the structural similarity between two vertices becomes large when they share many common structural neighbors.

In this paper, we take the neighborhood of vertices as the clustering criterion. We define three metrics for each vertex: the local density, the dependent similarity and the dependent vertex. Through these metrics, we can get the density peaks, noise vertices and clusters.

**Definition 3 (Local Density).** *Given  $\mu_{uv} = \frac{\sigma_t}{\sigma(u, v)}$ , the local density of  $u$ , denoted by  $\rho_u$ , is defined as the sum of the normal distribution values of  $\mu_{uv}$ , where  $\sigma_t$  is a default parameter called similarity threshold and  $v$  is the structural neighbor of  $u$ . That is*

$$\rho_u = \sum_{v \in N[u]} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\mu_{uv}^2}{2}\right) \quad (2)$$

As a rule of thumb, the value of  $\sigma_t$  is around 20% of the structural similarities in descending order. Intuitively, the local density becomes smaller with the decrease of  $\sigma(u, v)$ , which conforms to the general law.

**Definition 4 (Dependent Similarity).** *The dependent similarity of  $u$ , denoted by  $\delta_u$ , is defined as the maximum structural similarity between  $u$  and the neighbor with higher density. That is*

$$\delta_u = \max_{v: \rho_v > \rho_u, v \in N(u)} (\sigma(u, v)) \quad (3)$$

Note that, the open neighborhood of  $u$  doesn't include  $u$  itself. If there is no vertex with higher density in the neighborhood of  $u$ , we set  $\delta_u = 0$ .

**Definition 5 (Dependent Vertex).** *The dependent vertex of  $u$ , denoted by  $\varphi_u$ , is defined as  $u$ 's the most similar neighbor with higher density. That is*

$$\varphi_u = \operatorname{argmax}_{v: \rho_v > \rho_u, v \in N(u)} (\sigma(u, v)) \quad (4)$$

According to Definitions 4 and 5, we know that  $\delta_u$  is the structural similarity between  $u$  and  $\varphi_u$ . If there are two or more dependent vertices of  $u$ , we randomly pick one among them.

A vertex  $u$  with a small  $\rho_u$  is called a *noise vertex* [7], whose density is not bigger than a predefined value  $\xi$  (i.e.,  $\rho_u \leq \xi$ ). For a vertex with a large  $\rho$ , a relatively small  $\delta$  implies that vertex  $u$  is the density peak of its own structural neighborhood, since it is dissimilar with higher density neighbors, or it has no

higher density neighbors (i.e.,  $\delta = 0$ ). Thus, in a graph, *density peaks* are the vertices with large  $\rho$  as well as small  $\delta$ . Note that, the non-noise vertices with  $\delta = 0$  must be density peaks since it has no higher density neighbors.

If  $\varphi_u = v$ , it is highly possible that  $u$  and  $v$  belong to the same cluster [16]. Inspired by [7], we say that vertex  $u$  is *structure – dependent* on  $v$ . For a set of vertices  $\{v_1, v_2, \dots, v_n\}$ , if there exists a dependent chain, such that  $v_i$  ( $1 \leq i \leq n - 1$ ) is structure-dependent on  $v_{i+1}$  and the end vertex  $v_n$  is not dependent on any other vertex, we say that  $v_n$  is  $v_i$ 's *dependent root* and  $v_i$  ( $1 \leq i \leq j - 1$ ) is *dependency – reachable* to  $v_j$  ( $i + 1 \leq j \leq n$ ), denoted by  $v_i \rightarrow v_j$ . Now, we define a cluster as structure-dependent vertices.

**Definition 6 (Structure-Dependent Cluster).** *A non-empty subset  $C \subseteq V$  is called a structure-dependent cluster such that:*

- (*Maximality*) *If a vertex  $u \in C$ , any non-noise vertex  $v$  that is dependency-reachable to  $u$  also belongs to  $C$ .*
- (*Traceability*) *For any vertices  $v_1, v_2, \dots \in C$ , they have the same dependent root, which is the density peak in  $C$ .*

For a vertex, it is either a member of a cluster, or it is a noise vertex. If a noise vertex  $u$  has neighbors belonging to two or more different clusters,  $u$  is a *hub*. Otherwise,  $u$  is an *outlier*.

The following theorem validates the correctness of our proposed algorithm. Given a graph  $G = (V, E)$ , we can find structure-dependent clusters in a two-step approach. First, a density peak is chosen from  $V$ . Second, we retrieve all the vertices that are dependency-reachable to the density peak to obtain the cluster.

**Theorem 1.** *Given  $u \in V$ , if  $u$  is a density peak, then the set of vertices, which consists of  $u$  and the non-noise vertices dependency-reachable to  $u$ , is a structure-dependent cluster.*

*Proof.* ( $C \neq \emptyset$ ) Given  $C \subseteq V$ ,  $u$  is a density peak and  $u \in C$ . Thus,  $C \neq \emptyset$ .

(*Maximality*) Given vertex  $p \in C$ , non-noise vertex  $q \in V$  and  $q \rightarrow p$ , if  $p = u$ ,  $q \rightarrow u$ . If  $p \neq u$ ,  $q \rightarrow p$  and  $p \rightarrow u$ . Since dependency reachability is transitive,  $q \rightarrow u$ . Thus,  $q \in C$ .

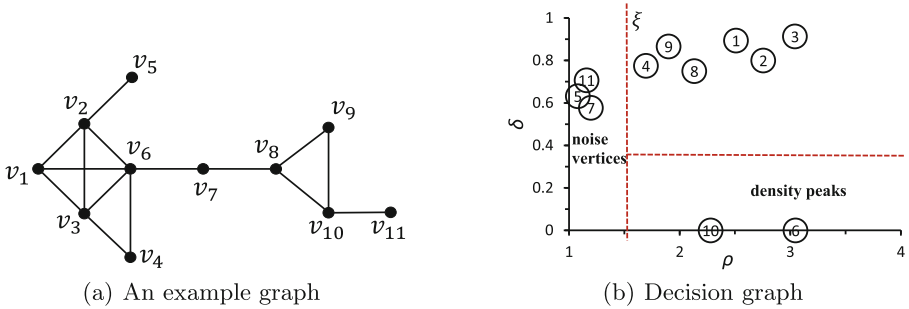
(*Traceability*) Let  $v_1, v_2, \dots, v_n \in C$  and the corresponding dependent roots are  $u_1, u_2, \dots, u_n$ . By means of reduction to absurdity, we can assume that  $\exists w_1, w_2 \in \{u_1, u_2, \dots, u_n\}$  and  $w_1 \neq w_2$ . Since  $w_1 \neq w_2$ ,  $\exists p \in \{w_1, w_2\}$  and  $p \neq u$ . By assumption,  $p \rightarrow u$ . Since  $p$  is the dependent root and not dependent on any other vertex,  $u_1 = u_2 \dots = u_n = u$ . Thus, any vertices in  $C$  have the same dependent root, which is the density peak.

### 3.2 DPSCAN

In this section, we describe the algorithm DPSCAN which can detect clusters, hubs and outliers. As mentioned in Subsect. 3.1, we first find all the density peaks

**Table 1.** Advantages of DPSCAN when compared with GS\*-Query, LPA, Modularity and GN.

Algorithms	# of clusters	Parameter tuning	Noise detection	Time complexity
DPSCAN	Flexible	No	Yes	$O(\sum_{c \in \mathbb{C}}  V_C )$
GS*-Query [22]	Automatic	Yes	Yes	$O(\sum_{c \in \mathbb{C}}  E_C )$
LPA [14]	Automatic	No	No	$O( V  +  E )$
Modularity [3]	Automatic	No	No	$O( E  +  V  \log^2  V )$
GN [6]	Manual	No	No	$O( V  E ^2)$


**Fig. 1.** An example of DPSCAN

and noise vertices. Then, we obtain the clusters grown from the density peaks. Finally we identify the noise vertices as bridges or outliers.

According to Definitions 3 and 4, we can generate a *decision graph* by taking  $\rho$  as  $x$  axis and  $\delta$  as  $y$  axis. The identification of noise vertices and density peaks is accomplished through interaction with the decision graph [16]. Based on the decision graph, the selection of density peaks is flexible. According to Definition 6, the number of density peaks determines the number of clusters. Thus, users can select the number of clusters according to the requirements (e.g. [13]). If users need to automatically get clusters (e.g. [4]), they only need to select all density peaks.

To illustrate the process of DPSCAN algorithm, we use the graph  $G$  in Fig. 1(a) to give the example. Figure 1(b) shows the decision graph of the graph  $G$ . By observing the decision graph, the noise vertices can be identified in the left region with  $\rho \leq \xi$ . There are three noise vertices:  $V_n = \{v_5, v_7, v_{11}\}$ . Then, the vertices  $v_6$  and  $v_{10}$  with large  $\rho$  as well as small  $\delta$  in the bottom right region are identified as density peaks. Then, we explore remaining vertices in non-increasing order of their local densities, and assign each remaining vertex and its dependent vertex to the same cluster. We get the clusters:  $\mathbb{C} = \{\{v_1, v_2, v_3, v_4, v_6\}, \{v_8, v_9, v_{10}\}\}$ . Finally, for the vertices in  $V_n$ ,  $v_7$  is identified as a hub vertex since its neighbors  $v_6$  and  $v_8$  belong to different clusters, and  $v_5$  and  $v_{11}$  are outliers since they have only one neighbor.

**Algorithm 1.** DPSCAN**Input:** a graph  $G(V, E)$ **Output:** the clusters, hubs and outliers in  $G$ 


---

```

1:  $V_p \leftarrow$  density peaks selected from decision graph;
2:  $V_n \leftarrow$  noise vertices selected from decision graph;
3: Set  $\forall u \in V$  as unexplored ( $u.explored = false$ );
4: for  $u \in V_n$  do
5:    $u.label = noise$ ;
6:    $u.explored = true$ ;
7: end for
8: for  $u \in V_p$  do
9:   generate a new clusterID  $i$ ;
10:   $u.clusterID = i$ ;
11:   $u.explored = true$ ;
12: end for
13: for each unexplored vertices  $u \in V$  in non – increasing order w.r.t.  $\rho_u$  do
14:   $u.clusterID = \varphi_u.clusterID$ ;
15:   $u.explored = true$ ;
16: end for
17: for  $u \in V_n$  do
18:  if  $u$ 's neighbors belong to two or more different clusters then
19:     $u.label = hub$ ;
20:  else
21:     $u.label = outlier$ ;
22:  end if
23: end for

```

---

The pseudocode of DPSCAN is shown in Algorithm 1. Firstly, we select density peaks and noise vertices from the decision graph and initialize the clusters (lines 1–12). Then, the unexplored vertices and their dependent vertices are assigned to the same clusters (lines 13–16). We retrieve unexplored vertices in non-increasing order of their local densities, since the dependent vertex of each vertex should have been clustered in advance. Finally, the noise vertices are further labeled with hubs and outliers (lines 17–23).

The time complexity for calculating all structural similarities is  $O(\alpha(G) \cdot m)$ , which has been discussed in [2].  $\alpha(G)$  is the arboricity of  $G$  and  $m$  is the number of edges. It costs  $O(m \log(m))$  time to sort similarities and get the similarity threshold. Then, it costs  $O(m)$  time calculating local densities, dependent vertices and dependent similarities, since they entail the retrieval of all the neighbors of vertices. Thus time complexity of initialization phase of DPSCAN is  $O(m \cdot (\alpha(G) + \log(m)))$ . For Algorithm 1, ignoring the time of interaction with the decision graph, it costs  $O(n)$  time to explore noise vertices and density peaks in lines 4–12. Here,  $n$  is the number of vertices. It costs  $O(n \log(n))$  time to sort vertices, and the time complexity for exploring unexplored vertices (lines 13–16) is  $O(n)$ . Thus, the total time complexity of Algorithm 1 is  $O(n \log(n))$ .

## 4 DP-Index

DPSCAN is flexible, which can online-compute any number of reasonable clusters. Sometimes, users may need to run an algorithm several times according to different requirements. However, for any vertex  $u \in V$ , the local density, the dependent vertex and the dependent similarity of  $u$  are fixed. For big graphs, this may consume considerable time.

---

**Algorithm 2.** DP-Index

---

**Input:** a graph  $G(V, E)$

**Output:** DP-Index of  $G$

```

1: for each edge  $(u, v) \in E$  do
2:   if  $\sigma(u, v) = \emptyset$  then
3:     compute  $\sigma(u, v)$  based on Eq. 1;
4:      $\sigma(v, u) \leftarrow \sigma(u, v)$ ;
5:   end if
6: end for
7:  $\mathfrak{S} \leftarrow \emptyset$ ;
8: for each  $u \in V$  do
9:   compute  $\rho_u$  based on Eq. 5;
10:  compute  $\delta_u, \varphi_u$  based on Eq. 3 and Eq. 4, respectively;
11:   $\mathfrak{S}_u \leftarrow \{\rho_u, \delta_u, \varphi_u\}$ ;
12:   $\mathfrak{S} \leftarrow \mathfrak{S} \cup \{\mathfrak{S}_u\}$ ;
13: end for
14: sort vertices in  $\mathfrak{S}$  in non-increasing order of their local densities;
15: return  $\mathfrak{S}$ ;

```

---

To solve this problem, we propose a novel index named **DP-Index** to solve this problem. The main idea for our index structure is the maintenance of the three metrics for each vertex, which are the local density, the dependent vertex and the dependent similarity. Based on the index, we can online-compute reasonable clusters efficiently. Given the three metrics of each vertex, the result is easily obtained by scanning the vertices following the same procedures as detailed in Algorithm 1.

DP-Index contains the vertices in  $G$  in addition to the three metrics of each vertex. To speed up the construction of DP-Index, we further propose two optimization techniques.

**Optimization of Local Density.** In Definition 3, we take the Standard normal distribution as the weight of  $\mu_{uv}$ . Therefore, it follows Pauta criterion. For  $N(\mu, \sigma^2)$ , the probability of  $\mu_{uv}$  distributed in  $(\mu - 2\sigma, \mu + 2\sigma)$  is 0.9545. Thus, in this paper, we set  $0 < \mu_{uv} \leq 2$  to filter the small structural similarities between  $u$  and its structural neighbors. Since DPSCAN is only sensitive to the relative magnitude of local density in different vertices, we simplify the local density

formula and take the kernel of the Normal distribution. Consequently, the local density of a vertex  $u$  is

$$\rho_u = \sum_{v \in N[u], 0 < \mu_{uv} \leq 2} \exp\left(-\frac{\mu_{uv}^2}{2}\right) \quad (5)$$

**Optimization of Structural Similarity Computation.** In the initialization phase of DPSCAN, the structural similarity between vertices  $u$  and  $v$  is computed twice: one in the direction  $\sigma(u, v)$  when exploring edge  $(u, v)$  and the other in the direction  $\sigma(v, u)$  when exploring edge  $(v, u)$ . Thus, we propose a pruning rule to speed up the structural similarity computation. When exploring edge  $(u, v)$ , we assign the similarity  $\sigma(u, v)$  to  $\sigma(v, u)$ , which can directly reduce structural similarity computations by half.

The pseudocode for the construction of DP-Index based on two optimization techniques is shown in Algorithm 2. We first compute the similarities for every pair of adjacent vertices in lines 1–6. Then we compute the three metrics and construct DP-Index in lines 7–13. Finally, the vertices are sorted in non-increasing order of their local densities (line 14).

The space cost of DP-Index is  $O(n)$ . The time complexity of Algorithm 2 is  $O(m \cdot (\alpha(G) + \log(m)) + n \log(n))$ , which has been discussed in Subsect. 3.2. Since the vertices have been sorted in DP-Index, the time complexity of Algorithm 1 based on DP-Index is  $O(n)$ , which is only proportional to the size of vertices.

**Computing All Clusters.** Based on DP-Index, DPSCAN can be naturally extended to compute the set  $\mathbb{C}$  of all clusters. The noise vertices whose local densities are not bigger than  $\xi$  ( $\xi \geq 0$ ) are firstly filtered during the construction of DP-Index in Algorithm 2. Then, after obtaining the density peaks through the decision graph, the execution procedure is the same as that in lines 8–16 of Algorithm 1. To compute the set  $\mathbb{C}$  of all clusters, the time complexity is  $O(\sum_{C \in \mathbb{C}} |V_C|)$ , which is only dependent on the result size of vertices excluding outliers and hubs. Here,  $\mathbb{C}$  is the result set of all clusters and  $|V_C|$  is the number of vertices in a specific cluster  $C$ .

## 5 Experiments

In this section, we evaluate our proposed algorithm DPSCAN on synthetic as well as real-world networks to demonstrate its benefits.

**Selection of Comparison Methods.** To evaluate the performance of DPSCAN, we compare it to the state-of-the-art structural graph clustering algorithm. In addition, to sufficiently verify the effectiveness of our method, we compare with four representatives of other graph clustering algorithms.

- **GS\*-Query** [22] is the state-of-the-art algorithm for structural graph clustering, which is an index-based approach of SCAN [23].

- **FluidC** [13] is a recently proposed graph clustering approach, which is based on the idea of fluids interacting in an environment.
- **SSLPA** [4] is a very fast parallel propagation-based graph clustering approach for large networks. This method combines the advantages of both the synchronous and asynchronous models of LPA [14].
- **Modularity** [3] is a popular graph clustering algorithm based on the modularity measure, which begins with each node in its own cluster and joins the pair of clusters that can increase the modularity most.
- **Girvan-Newman** [6] (in the following named GN) is a well-known graph clustering algorithm based on the betweenness of the edges in the network. The algorithm finds the optimal partitions by gradually removing the edge with the highest betweenness from the network.

For all experiments, without further statement, FluidC and GN specify the cluster number  $K = |C|$ , where  $|C|$  is the true number of clusters of the network. GS\*-Query takes the parameters of the best performance in experiments. All experiments have been performed on a workstation with 2.9 GHz CPU and 8.0 GB RAM.

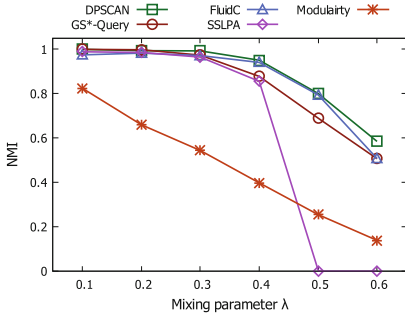
**Evaluation Measures.** To extensively compare the effectiveness of different graph clustering algorithms, the clustering performance is directly measured by two widely used evaluation measures: Normalized Mutual Information (NMI) [20] and Adjusted Rand Index (ARI) [15].

## 5.1 Synthetic Networks

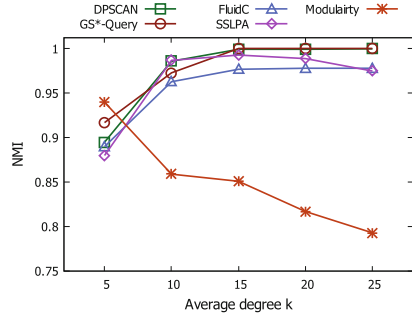
In this section, we generate several synthetic networks to compare the performance of various graph clustering algorithms. The LFR benchmark network [9] is applied, where the degree distribution and the cluster size can be easily adjusted. Due to the high time complexity of GN, we limit the comparison to the clustering algorithms DPSCAN, GS\*-Query, SSLPA, FluidC and Modularity.

**Inter-Cluster Edge:** First, we evaluate how the algorithms respond to the networks by varying the inter-cluster edges. We fix the node average degree and change the mixing parameter  $\lambda$  from 0.1 to 0.6 to generate networks with different inter-cluster edges. All networks consist of 5000 nodes with the average degree  $k = 15$ . With the increase of the mixing parameter, the performance of all five approaches measured by NMI is shown in Fig. 2(a). In Fig. 2(a), we can see Modularity is more sensitive to inter-edges, and the performance is not comparable with the other four algorithms. Regarding the SSLPA, its performance is satisfactory at the beginning and starts to decrease dramatically as soon as more inter-edges are added (with  $\lambda = 0.4$ ). For structural clustering algorithms, DPSCAN and GS\*-Query show a similar trend. DPSCAN, GS\*-Query and FluidC almost achieve the perfect clusterings with the mixing parameter up to 0.4, and begin to decrease with adding more noise edges into the graph data. DPSCAN achieves relatively better results than that of FluidC and GS\*-Query.





(a) Varying the number of inter-cluster edges in the graph.



(b) Varying the densities of clusters in the graph.

**Fig. 2.** Performance of different algorithms on the LFR benchmark networks.

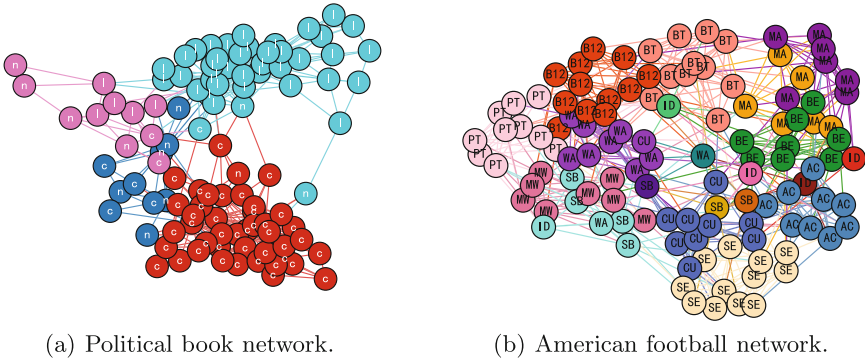
**Table 2.** Performance of different graph clustering algorithms on real-world data sets.

	Karate		Polbooks		Football		Email	
	ARI	NMI	ARI	NMI	ARI	NMI	ARI	NMI
DPSCAN	<b>0.882</b>	<b>0.837</b>	<b>0.695</b>	<b>0.593</b>	<b>0.888</b>	<b>0.942</b>	<b>0.500</b>	<b>0.709</b>
GS*-Query	0.725	0.684	0.673	0.581	0.861	0.929	0.218	0.708
FluidC	<b>0.882</b>	<b>0.837</b>	0.635	0.547	0.806	0.908	-	-
SSLPA	0.473	0.450	0.594	0.534	0.757	0.882	0.011	0.258
Modularity	0.680	0.707	0.638	0.531	0.476	0.720	0.172	0.503
GN	<b>0.882</b>	<b>0.837</b>	0.680	0.576	0.880	0.937	0.001	0.131

**Cluster Density:** Furthermore, we evaluate the algorithms on LFR benchmark graphs with different average degrees, which is called cluster density [17]. Here we fix  $\lambda = 0.1$ , and change the average degree  $k$  from 5 to 25. All networks consist of 5000 nodes. In Fig. 2(b), we can see that DPSCAN, FluidC, SSLPA and GS\*-Query achieve perfect clusterings for all graphs. DPSCAN and GS\*-Query perform a bit better than FluidC and SSLPA. For Modularity, the performance is better than the other algorithms with a low cluster density ( $k = 5$ ) and becomes worse with the increase of the average degree.

### 5.2 Real World Data

We use four publicly available real-world networks to evaluate different graph clustering algorithms. All networks and corresponding detailed descriptions can be found in UCI network data repository (<https://networkdata.ics.uci.edu/index.php>) and Stanford large network dataset collection (<https://snap.stanford.edu/data/index.html>).



**Fig. 3.** Performance of DPSCAN on real world networks.

**Zachary’s Karate Club Network:** The famous social network reflects the friendships between 34 members of a karate club at a US university in the 1970s. The network could be divided into two clusters, which reflects the disagreement between the administrator and the instructor.

The performance of the different graph clustering algorithms is summarized in Table 2. For DPSCAN, FluidC and GN, they yield comparable results and identify the clusters with a high degree of success. For GS\*-Query, SSLPA and Modularity, they yield relatively low-quality results.

**Books About US Politics:** This network of books is about US politics published around the time of the 2004 presidential election. Nodes represent books sold by the online bookseller Amazon.com. Edges between books represent frequent co-purchasing of books by the same buyers. This network consists of 105 nodes and 441 edges. Mark Newman divides these books into three categories, which are “liberal”, “neutral”, and “conservative”.

Most books can be correctly clustered by DPSCAN with the highest quality (Table 2). However, DPSCAN divides these books into four clusters (Fig. 3(a)). The “neutral” books are divided into two clusters, since the density-based notion of clusters tends to find smaller clusters that are more closely connected [5]. For algorithms of GS\*-Query, FluidC, SSLPA, Modularity and GN, they produce comparable groupings on this network.

**American College Football:** The network derived from the American football games of the schedule of Division IA colleges during regular season Fall 2000, where vertices in the graph represent teams, and edges represent regular-season games between the two teams they connect. This network consists of 115 nodes and 613 edges. The teams are divided into 11 conferences. In addition, there are five independent teams (labeled with ‘ID’ in Fig. 3(b)), which play against other teams in many conferences but belong to none.

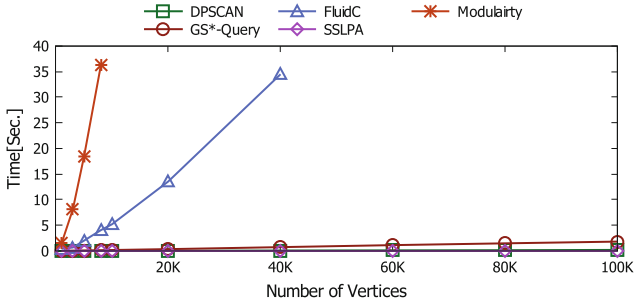


Fig. 4. The runtime of the different graph clustering algorithms.

DBSCAN identifies the clusters and noise vertices with the highest cluster quality compared to the other five approaches (ARI = 0.888, NMI = 0.942). Most of the teams are correctly grouped into corresponding clusters, and four correct independent teams are identified (Fig. 3(b)). The independent teams show strong properties as hubs, since they have edges that connect them to a large number of clusters. GS\*-Query takes the parameters as suggested by authors [23]. GS\*-Query also finds the similar cluster structure as DPSCAN and identifies three correct independent teams. FluidC, SSLPA and GN also perform well, but they cannot identify hubs. For Modularity, it is difficult to discover the natural clusters (Table 2).

**Email-Eu-core:** This network consists of 1005 nodes and 25571 edges, which is generated using email data from a large European research institution. Each node represents an individual, which belongs to exactly one of 42 departments at the research institute. There is an edge  $(u, v)$  in the network if person  $u$  sent person  $v$  at least one email.

Since FluidC is designed to run on connected graphs, it cannot handle this network. DPSCAN obtains the highest cluster quality compared to the other four approaches (ARI = 0.500, NMI = 0.709). GS\*-Query and Modularity also perform well (Table 2). However, for SSLPA and GN, they fail to discover the clusters.

### 5.3 Runtime

To assess the scalability of DPSCAN with respect to network size, we generate several benchmark networks [9] with different node sizes ranging from  $10^3$  to  $10^5$  (corresponding edge sizes ranging from  $10^4$  to  $10^6$ ), by fixing the average node degree  $k = 15$  and the mixing parameter  $\lambda = 0.1$ . Due to the time complexity of GN, we limit the comparison to the clustering algorithms DPSCAN, GS\*-Query, SSLPA, FluidC and Modularity. To obtain more accurate runtime results, each method is executed 10 times and the averaged time is calculated. Figure 4 shows the running time for different graph clustering algorithms.

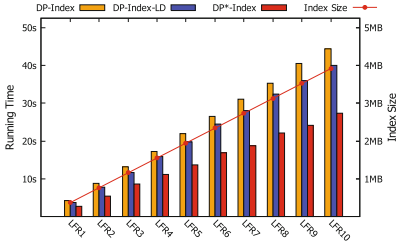


Fig. 5. Index size and time cost for different graphs.

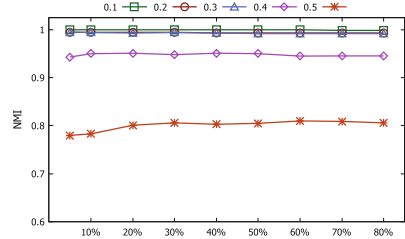


Fig. 6. The performance of DPSCAN on the LFR benchmark networks by varying similarity threshold  $\sigma_t$ .

For the structural clustering algorithm, DPSCAN is faster than GS\*-Query in our evaluation. In theory, to compute the set  $\mathcal{C}$  of all clusters in  $G$ , the time complexity of GS\*-Query based on GS\*-Index is  $O(\sum_{c \in \mathcal{C}} |E_C|)$  [22], which is on the result size of edges. The time complexity of DPSCAN based on DP-Index is  $O(\sum_{c \in \mathcal{C}} |V_C|)$ , which is on the result size of vertices. Normally, the number of edges is much greater than the number of nodes, especially for big graphs. Thus, it is demonstrated experimentally and theoretically that DPSCAN is faster than the state-of-the-art approach.

For other clustering algorithms, we can observe that DPSCAN is faster than Modularity and FluidC. Besides, DPSCAN is competitive with scalable parallel graph clustering algorithm SSLPA. Although SSLPA is a bit faster than DPSCAN, SSLPA suffers in the quality of resulting clusters.

### 5.4 Performance of DP-Index Construction

In this section, we evaluate the performance of DP-Index construction. We generate ten LFR graphs, LFR1, ..., LFR10, with node sizes ranging from  $10^4$  to  $10^5$ . To evaluate the effect of our two optimization techniques, we compare DP\*-Index with DP-Index and DP-Index-LD (Fig. 5). DP-Index-LD is obtained from DP-Index by adding the optimization of local density, while DP\*-Index integrates both two optimizations. In Fig. 5, the time cost of DP-Index gradually grows when the node number increases. We can see that each of the two optimization techniques in Sect. 4 can speed up the construction of the DP-Index for different graphs. The size of DP-Index for different graphs is also reported in Fig. 5. We can see that the size of DP-Index is proportional to the number of vertices. The space usage of DP-Index can be well bounded, and the DP-Index only costs 4MB in LFR10 with  $10^5$  vertices.

### 5.5 Analysis of Similarity Threshold

To evaluate the influence of similarity threshold  $\sigma_t$  on clustering results, we generate several benchmark networks [9] with different mixing parameters ranging

from 0.1 to 0.5, by fixing the average node degree  $k = 15$ . Assuming that the value of  $\sigma_t$  is around  $m\%$  of the structural similarities in descending order, Fig. 6 shows the performance of DPSCAN on the LFR benchmark networks by varying similarity threshold  $\sigma_t$ . In Fig. 6, we can observe that the results of DPSCAN are robust with respect to the choice of  $\sigma_t$ . The value of  $\sigma_t$  with small  $m\%$  will slightly affect the performance, since small structural similarities are filtered in Eq. 5. For  $\lambda = 0.5$ , with the increase of  $m\%$ , NMI slowly reaches the maximum value when  $m\% \approx 20\%$ . Then, the performance tends to stabilize. However, the density peaks are more difficult to be distinguished in the decision graph with the increase of  $m\%$ . Thus, the reasonable interval of  $m\%$  is from 20% to 40%. In this paper, we take 20% as the default value.

## 6 Conclusions

In this paper, we introduce DPSCAN, a novel structural graph clustering algorithm based on density peaks. Firstly, we propose three new metrics and the notion of structure-dependent cluster. Through theoretical analysis, we validate the correctness of our proposed algorithm. Then, a flexible graph clustering algorithm DPSCAN is proposed, which can identify clusters, hubs and outliers. To improve the performance of our approach, we further propose DP-Index and two optimization techniques. By using the DP-Index structure, the cluster results can be computed very quickly, which is only proportional to the size of vertices. Our extensive experiments demonstrate that DPSCAN outperforms the state-of-the-art structural graph clustering methods.

**Acknowledgements.** This work is supported by the National Key R&D Program of China (2018YFB1003404), the National Nature Science Foundation of China (61872070, U1435216, U1811261 and 61602103) and the Fundamental Research Funds for the Central Universities (N171605001).

## References

1. Bai, X., Yang, P., Shi, X.: An overlapping community detection algorithm based on density peaks. *Neurocomputing* **226**, 7–15 (2017)
2. Chang, L., Li, W., Qin, L., Zhang, W., Yang, S.: pSCAN: fast and exact structural graph clustering. *IEEE Trans. Knowl. Data Eng.* **29**(2), 387–401 (2017)
3. Clauset, A., Newman, M.E., Moore, C.: Finding community structure in very large networks. *Phys. Rev. E* **70**(6), 066111 (2004)
4. Cordasco, G., Gargano, L.: Community detection via semi-synchronous label propagation algorithms. In: 2010 IEEE International Workshop on Business Applications of Social Network Analysis (BASNA), pp. 1–8. IEEE (2010)
5. Falkowski, T., Barth, A., Spiliopoulou, M.: Studying community dynamics with an incremental graph mining algorithm. In: AMCIS 2008 Proceedings, p. 29 (2008)
6. Girvan, M., Newman, M.E.: Community structure in social and biological networks. *Proc. Natl. Acad. Sci.* **99**(12), 7821–7826 (2002)
7. Gong, S., Zhang, Y., Yu, G.: Clustering stream data by exploring the evolution of density mountain. *Proc. VLDB Endowment* **11**(4), 393–405 (2017)

8. Huang, J., Sun, H., Han, J., Deng, H., Sun, Y., Liu, Y.: Shrink: a structural clustering algorithm for detecting hierarchical communities in networks. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, pp. 219–228. ACM (2010)
9. Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. *Phys. Rev. E* **78**(4), 046110 (2008)
10. Lim, S., Ryu, S., Kwon, S., Jung, K., Lee, J.G.: LinkSCAN\*: overlapping community detection using the link-space transformation. In: 2014 IEEE 30th International Conference on Data Engineering (ICDE), pp. 292–303. IEEE (2014)
11. Newman, M.E.: Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E* **74**(3), 036104 (2006)
12. Onizuka, M., Fujimori, T., Shiokawa, H.: Graph partitioning for distributed graph processing. *Data Sci. Eng.* **2**(1), 94–105 (2017)
13. Parés, F., et al.: Fluid communities: a competitive, scalable and diverse community detection algorithm. In: Cherifi, C., Cherifi, H., Karsai, M., Musolesi, M. (eds.) COMPLEX NETWORKS 2017. SCI, vol. 689, pp. 229–240. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-72150-7\\_19](https://doi.org/10.1007/978-3-319-72150-7_19)
14. Raghavan, U.N., Albert, R., Kumara, S.: Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E* **76**(3), 036106 (2007)
15. Rand, W.M.: Objective criteria for the evaluation of clustering methods. *J. Am. Stat. Assoc.* **66**(336), 846–850 (1971)
16. Rodriguez, A., Laio, A.: Clustering by fast search and find of density peaks. *Science* **344**(6191), 1492–1496 (2014)
17. Shao, J., Han, Z., Yang, Q., Zhou, T.: Community detection based on distance dynamics. In: Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1075–1084. ACM (2015)
18. Shi, J., Malik, J.: Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(8), 888–905 (2000)
19. Shiokawa, H., Fujiwara, Y., Onizuka, M.: SCAN++: efficient algorithm for finding clusters, hubs and outliers on large-scale graphs. *Proc. VLDB Endowment* **8**(11), 1178–1189 (2015)
20. Strehl, A., Ghosh, J.: Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.* **3**, 583–617 (2002)
21. Sun, H., Huang, J., Han, J., Deng, H., Zhao, P., Feng, B.: gSkeletonClu: density-based network clustering via structure-connected tree division or agglomeration. In: 2010 IEEE 10th International Conference on Data Mining (ICDM), pp. 481–490. IEEE (2010)
22. Wen, D., Qin, L., Zhang, Y., Chang, L., Lin, X.: Efficient structural graph clustering: an index-based approach. *Proc. VLDB Endowment* **11**(3), 243–255 (2017)
23. Xu, X., Yuruk, N., Feng, Z., Schweiger, T.A.: SCAN: a structural clustering algorithm for networks. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 824–833. ACM (2007)
24. Zhou, K., Martin, A., Pan, Q., Liu, Z.: SELP: semi-supervised evidential label propagation algorithm for graph data clustering. *Int. J. Approximate Reasoning* **92**, 139–154 (2018)
25. Zhou, K., Pan, Q., Martin, A.: Evidential community detection based on density peaks. In: Destercke, S., Denoeux, T., Cuzzolin, F., Martin, A. (eds.) BELIEF 2018. LNCS (LNAI), vol. 11069, pp. 269–277. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-99383-6\\_33](https://doi.org/10.1007/978-3-319-99383-6_33)



# Efficient Processing of Spatial Group Preference Queries

Zhou Zhang<sup>1</sup>, Peiquan Jin<sup>1,2</sup>(✉), Yuan Tian<sup>1</sup>, Shouhong Wan<sup>1,2</sup>,  
and Lihua Yue<sup>1,2</sup>

<sup>1</sup> University of Science and Technology of China, Hefei, China  
jpeq@ustc.edu.cn

<sup>2</sup> Key Laboratory of Electromagnetic Space Information,  
Chinese Academy of Sciences, Hefei, China

**Abstract.** POIs (points of interest) as well as users' check-in information and their ratings on POIs have been widely studied in systems providing location-based services. We note that users usually have their own preferences for POI categories and their own network of friends. Therefore, we aim to provide for a group of users a new kind of POI-finding query that considers not only POI preferences of each user but also other aspects of location-based social networks such as users' locations and POI ratings. We name such a new query as Spatial Group Preference (SGP) query. For a group of users, an SGP query returns top- $k$  POIs that are much likely to satisfy the needs of users. Specially, we propose a new evaluation model that considers user preferences for user preferences for POI categories. Based on this model, we develop basic algorithms based on R-tree to evaluate SGP queries. Further, we design a new index structure called CR-tree to accelerate the query performance. We prove that CR-tree has better pruning efficiency than the traditional R-tree. We conduct experiments on a simulation dataset as well as two real datasets with respect to various configurations. The results suggest the efficiency of our proposal.

**Keywords:** Location-based service · Group preference · CR-tree

## 1 Introduction

Recently, with the popularity of GPS-enabled smart phones, location-based social network becomes a hot topic. Location-based social networks allow a group of users (circle of friends) to share their location information each other. Together with other information about POIs (points of interest) such as POI locations, POI categories (e.g., restaurant), and POI ratings, we can provide a variety of services for people.

This paper studies an interesting type of query called *Spatial Group Preference* (SGP) query. Given a set of POIs and a group of users, an SGP query retrieves  $k$  POIs that are expected to satisfy the overall need of the group of users. We assume that each user in a group has his or her current location and a preference list for POI categories, e.g., restaurant, theater, and shopping mall. Figure 1 shows an example of SGP queries, where Tony, Jack and Cindy want to find a restaurant for dinner and have some other activities around the restaurant after dinner. This query is actually an SGP query where  $k = 1$ .

Suppose that Tony’s preference list is {restaurant (0.3), moive theater (0.6), café (0.1)}, where each element in the list indicates a POI category and its weight according to Tony’s decision. Similarly, Jack’s preference list is {restaurant (0.2), cafe (0.4), bar (0.4)} and Cindy’s is {restaurant (0.6), cafe (0.4)}. Here, the problem is how to determine the best candidate for the SGP query. The table embedded in Fig. 1 shows three possible answers to the query. If we simply sum the weights of each category in the preference lists of Tony, Jack and Cindy, we will select r1 as the candidate. If we only consider distance, the best answer is r2. If we consider both weights and distances, the best candidate is r3.

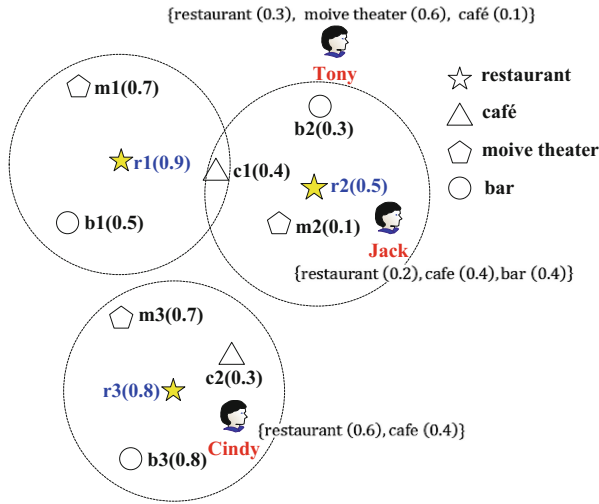


Fig. 1. An example of SGP queries (“Finding the best-fit restaurant for Tony, Jack and Cindy”)

To the best of our knowledge, so far there are no previous works that can directly answer SGP queries. In this paper, we present a framework for processing SGP queries. We first give the formal definition for SGP queries and then present several basic R-tree-based algorithms for answering a SGP query. Further, we propose a new index structure called CR-tree and develop the CR-tree-based efficient algorithm for SGP queries. We theoretically analyze the pruning efficiency of the CR-tree, and experimentally demonstrate its efficiency. Briefly, we make the following contributions in this paper:

- (1) We define a new kind of query called SGP query for location-based social networks, and propose a new evaluation model that considers user preferences for POI categories, including locations and ratings, and the mutual influence between POIs. Compared with existing approaches, this model is user-aware and can provide more reasonable ranking for POIs.
- (2) We propose a new index structure called CR-tree and the corresponding pruning strategy to improve the query performance of SGP queries. We theoretically prove the upper bound of the pruning strategy, and demonstrate that it is more efficient than the traditional R-tree.



- (3) We conduct experiments on a simulation dataset as well as on two real datasets with respect to various configurations. A number of competitor algorithms are compared with the proposed CR-tree-based algorithm. The results suggest the efficiency of our proposal.

## 2 Problem Definition and Related Work

We assume that there are  $m$  categories of POIs, which are represented as  $C = \{c_1, c_2, c_3, c_4, \dots, c_m\}$ . Each  $c_i (1 \leq i \leq m)$  represents a category. A set of POIs is represented by  $P = \{p_1, p_2, p_3, p_4, \dots, p_l\}$ . Each  $p_i (1 \leq i \leq l)$  is represented by a triple  $\langle p_i.loc, p_i.t, s(p_i) \rangle$ . Here,  $p_i.loc$  is the location of  $p_i$ ;  $p_i.t$  is the category associated with  $p_i$  and  $s(p_i)$  is the rating of  $p_i$ , which is a value between 0 and 1 and can be obtained from social network platforms.

**Definition 1 (SGP query).** Given a set of POIs  $P$ , a group of querying users  $Q = \{u_1, u_2, u_3, \dots, u_n\}$ , a targeted category  $c$ , and an integer  $k$ , a Spatial Group Preference (SGP) query retrieves a set  $S \subseteq P$  that consists of  $k$  POIs, such that:

- (1)  $(\forall x)(x \in S) \rightarrow x.t = c$
- (2)  $(\forall x)(\forall y)(x \in S \wedge y \in P - S \wedge y.t = c) \rightarrow sd(x, Q) \geq sd(y, Q)$

Here,  $sd(x, Q)$  returns the satisfaction degree between  $x$  and  $Q$ . ■

$Q = \{u_1, u_2, u_3, \dots, u_n\}$  represents a set of users and each user  $u_i$  is denoted as a tuple  $\langle u_i.loc, u_i.CW \rangle$ , where  $u_i.loc$  is the current location of user  $u_i$  and  $u_i.CW$  represents the preferences for POI categories of the user. To quantify the preferences, we assign a weight to each preferred POI category for each user. Consequently, we have  $u_i.CW = \{\langle u_i.c_1, u_i.w_1 \rangle, \langle u_i.c_2, u_i.w_2 \rangle, \dots, \langle u_i.c_{m'}, u_i.w_{m'} \rangle\}$ . Each tuple in  $u_i.CW$ , e.g.,  $\langle u_i.c_1, u_i.w_1 \rangle$ , means that the category  $c_1$  has a weight of  $w_1$ . We assume that  $0 \leq u_i.w_j \leq 1$ , yielding  $\sum_{j=1}^{m'} u_i.w_j = 1$ . In addition, we use  $C_Q^{rad}$  to represent all the categories contained in  $Q$ , i.e.,  $C_Q^{rad} = \bigcup_{i=1}^n \bigcup_{j=1}^{m'} u_i.c_j$ .

The key issue of answering an SGP query is how to define the satisfaction degree model  $sd(x, Q)$ . We define two metrics, namely *distance relevance* and *preference relevance*, to measure the satisfaction degree.

**Definition 2 (Distance Relevance) [17].** Given a candidate POI  $p$  and an SGP query  $Q$ , the distance relevance between  $p$  and  $Q$  is defined by (1):

$$\delta(p, Q) = 1 - \frac{d(Q, p)}{D_{max} \cdot |Q|} \quad (1)$$

Here,  $d(Q, p)$  represents the sum of the distances of each user to  $p$ .  $D_{max}$  is the diagonal length of the MBR covered by all the involved POIs. ■

**Definition 3 (Preference Relevance).** Given a candidate POI  $p$  and an SGP query  $Q$ , the preference relevance between  $p$  and  $Q$  is defined by (2):

$$\tau(p, Q) = \sum_{c_i \in C_Q^{rad}} \frac{1}{|Q|} \cdot \frac{W(c_i) \cdot r_{c_i}^{rad}(p)}{1 + \frac{d(p, p_{c_i}^{rad}(p))}{rad}} \quad (2)$$

Here,  $p_{c_i}^{rad}(p)$  is a POI with category  $c_i$  and it falls within the circle centered by  $p$  and the radius is  $rad$ .  $r_{c_i}^{rad}(p)$  is the rating of  $p_{c_i}^{rad}(p)$ .  $W(c_i) = \sum_{j=1}^{|Q|} u_j \cdot w_i$  is the sum of the weights assigned by all the users in the group to category  $d(p, p_{c_i}^{rad}(p))$  is the distance between  $p$  and  $p_{c_i}^{rad}(p)$ . ■

The definition of preference relevance aims to balance the preference weights and network proximity of POIs. Such balancing strategy can be found in various areas. For example, a previous study [18] proposed to balance the textual relevance and network proximity when answering spatial keyword queries on road networks.

**Definition 4 (Satisfaction Degree).** Given a candidate POI  $p$  and an SGP query  $Q$ , the satisfaction degree of  $p$  with respect to  $Q$  is measured by (3):

$$sd(p, Q) = \alpha \cdot \delta(p, Q) + (1 - \alpha) \cdot \tau(p, Q) \quad (3)$$

Here,  $\alpha$  is a smoothing parameter that is used to balance distance relevance and preference relevance. ■

Note that when  $\alpha$  is set to 1, we only consider the distance relevance, which is similar to the approach to the GNN query [2, 3, 20]. When  $\alpha$  is set to 0, we only consider the preference relevance, which can be regarded as a variant of the top- $k$  spatial preference query [11]. Thus, by setting a specific value for  $\alpha$ , our satisfaction model can be adjusted to different scenarios.

There are some related works that are closely related with SGP query. In 2004, Papadias et al. first proposed the concept of group nearest neighbor (GNN) query [2] in spatial databases, which is to find a suitable gathering place for a group of users scattered throughout the Euclidean distance space. They established the R-tree index to organize candidate gathering points and further extended this approach to Aggregate Nearest Neighbor (ANN) query [3]. Both of them studied the group nearest neighbor query in Euclidean distance space, which is not suitable for road networks [4, 5] due to the differences between road networks and the Euclidean distance space. There are other researches [6–10] paying attention to improve the efficiency and to reduce the I/O cost of queries in Euclidean distance space or road network. However, all the above works do not consider group preferences.

In 2007, Yiu et al. proposed a new kind of query called top- $k$  spatial preference query [11], where a new index called aR-tree was presented to accelerate the query. This query returns top- $k$  candidate objects whose rankings are defined by the quality of

other objects around them. Further, the work in [12] proposed an improved scoring method based on textual similarity for getting candidate objects. The idea of considering the influence of near objects was also applied to road networks [13–16]. For example, in [16] the authors mapped distances and the scores of other kinds of objects surrounding the candidate object into a two-dimensional space based on distance and score, and then used the dynamic skyline method to reduce the number of candidate objects. However, existing works on spatial preference queries are not user-aware, meaning that they do not consider the user preferences for POIs.

In 2016, Miao Li et al. studied location-aware group preference queries [17] that are a combination of GNN query and group preference. They return top- $k$  sites that consist of those sites having the minimum distance to the locations of scattered uses in a group and the sites matching the group preference. However, this approach only uses distance to evaluate the group preference queries. It is not suitable for location-based social networks where POI ratings need to be considered in query evaluation.

All of the above spatial preference query correlations do not consider the spatial attributes of the user group. In our previous work [19], we presented the basic concept and developed basic algorithms for SGP queries, which will be briefly discussed in Sect. 3, while in this paper we made further improvements including the CR-tree index and associated query processing algorithms. In addition, we provided theoretical analysis and extended experiments in this paper.

### 3 Basic Algorithms for SGP Query Processing

#### 3.1 The Baseline Algorithm (BA)

We first propose a baseline algorithm (BA) for SGP queries. In this algorithm, we build an R-tree for the POIs in each category. R-tree [1] is one of the most popular and widely used data structures for indexing spatial objects. In this algorithm, we first traverse the R-tree for category  $c$  (denoted as  $R_c$ ) starting from the root. If the current visited node is a non-leaf node, we execute a recursive call to all of its child nodes. Otherwise (it is a leaf node), for each POI in the leaf node, we traverse each  $R_{c_i}$  ( $c_i \in C_Q^{rad}$ ) and compute the satisfaction degree of the POI. Finally, we return the top- $k$  POIs that are within category  $c$  and have the highest  $k$  satisfaction-degree values.

#### 3.2 The aR-Tree Based Pruning Algorithm (PA)

In this algorithm, we use the aR-tree [11] to index the POIs in each category. The aR-tree is similar to R-tree, except that it puts extra aggregation information in each node. Differing from [11] that appends to each node the sum of the ratings of each sub-node, in this paper we add to each node the maximal rating of POIs among the entire sub-tree rooted by the node.

**Algorithm 1.**  $PA(N, Q, c, k)$ 

**Input:**  $N$  is the root of aR-tree,  $Q$  is the set of querying users,  $c$  is a given category,  $k$  is the number of results.

**Output:**  $H$ , which is a max heap maintaining the selected  $k$  POIs.

```

1  If  $sd^+(N, Q) > kth\_bestvalue$ 
2  |   If  $N$  is a non-leaf node
3  |   |   Execute  $PA$  recursively for each child node of  $N$ ;
4  |   Else
5  |   |   For each entry  $e \in N$ 
6  |   |   |   For each  $c_i \in C_Q^{rad} \wedge c_i \neq c$  do
7  |   |   |   |    $Score\_compute(e, N, R_{c_i}.Root)$ ;
8  |   |   |   |   Compute  $sd(e, Q)$  and Update  $H$  and  $kth\_bestvalue$ ;
9  |   Return  $H$ ;

```

**Algorithm 1** shows the details of the PA algorithm. Note that the max rating of POIs are stored in each node in the tree.  $sd^+(N, Q)$  (Line 1) is the upper bound of  $sd(p, Q)$ , and  $kth\_bestvalue$  is a global variable indicating the  $k$ th highest satisfaction degree. The routine of  $Score\_compute(e, N, R_{c_i}.Root)$  is to traverse each  $R_{c_i}$  ( $c_i \in C_Q^r \wedge c_i \neq c$ ) and compute the preference relevance of  $\tau(p, Q)$  for  $c_i$ .  $H$  is a max heap that always maintains the  $k$  POIs with category  $c$  to be returned.

Compared with the BA algorithm, the PA algorithm reduces the number of candidate POIs by using two pruning conditions. However, as it builds an aR-tree for each category, it has to repeatedly traverse the aR-trees for different categories. To this end, it should be more efficient if we can build one index structure for the POIs with different categories. This motivates the CR-tree that we propose in Sect. 4.

### 3.3 The Optimized PA Algorithm (OPA)

In the original PA algorithm, we check all the candidate POIs in the leaf node of the aR-tree corresponding to the target aggregation type  $c$ , and sequentially traverse each semantic of the group position semantic preference list for each of the POIs. Corresponding POIs in the range, and calculate the user's satisfaction value according to (2). Assuming that there are  $h$  candidate POIs in the leaf node and  $f$  position semantic categories are involved in the preference list, the leaf nodes need to traverse  $f * h$  times in the aR-tree, which is computationally expensive. In fact, multiple candidate POIs in the same leaf node are similar in space, and their corresponding rad ranges often overlap. Therefore, we consider a leaf node as a whole unit, and perform one traversing in the preference list to obtain the POIs of each category in the range around the leaf node. At the same time, we calculate the user satisfaction value for each candidate POI in the node. This optimized PA algorithm is named OPA in the following texts.

Compared with the BA algorithm, the PA and OPA algorithms can greatly reduce the searching costs to reduce the number of the candidate POIs within the given category and the number of preferred POIs around the candidate POIs. However, since

the two algorithms respectively establish the corresponding aR-tree according to the POI category, the same query needs to traverse the location category index involved in the preference list in order to further improve the query efficiency.

## 4 CR-Tree-Based SGP Query Processing

### 4.1 Category R-Tree (CR-Tree)

In this section, we propose an efficient R-tree based index called CR-tree (*Category R-tree*) for SGP query acceleration. The CR-tree is an extension of the R-tree. Specially, we add additional information in the node of R-tree, including category MBRs and the max rating of POIs in each category MBR, which forms one index structure for all POIs with multiple categories. In CR-tree, both leaf and non-leaf nodes store their category and range information. Figure 2 shows an example of a leaf node in CR-tree. In this example, there are 9 POIs (a1, a2, a3, b1, b2, b3, c1, c2, c3) labeled with three categories (white circle, black circle, white triangle). Three category MBRs record the scopes and the max ratings of surrounding POIs labeled with corresponding categories. For example, for  $C_1$ , it records a range of  $\langle (0, 53), (23, 70) \rangle$  and the max rating is 0.8. The non-leaf nodes of the CR-tree contain entries of  $(ptrs, cmbr, max\ rating)$ , where  $ptr$  is a pointer to the child nodes;  $cmbr$  is the node category MBR that covers all the category MBRs in the child nodes;  $max\ rating$  is the max rating of all the POIs in  $cmbr$ .

Similar to the non-leaf nodes in R-tree, CR-tree's non-leaf nodes also store pointers to individual child nodes and MBRs that cover the MBRs of all child nodes. In addition, to ensure the hierarchy of Category MBR The Category MBR that covers the Category MBR of all child nodes in the same category is also included in each internal node, and the corresponding maximum score value is also stored in the hierarchy.

The indexing of spatial objects in CR-tree and R-tree is consistent, and it still maintains the R-tree hierarchy. Therefore, operations on the CR-tree, such as POI insertion and deletion are similar to R-tree operations, except that we add the operations of updating category MBRs and the max rating of POIs when adjusting the tree.

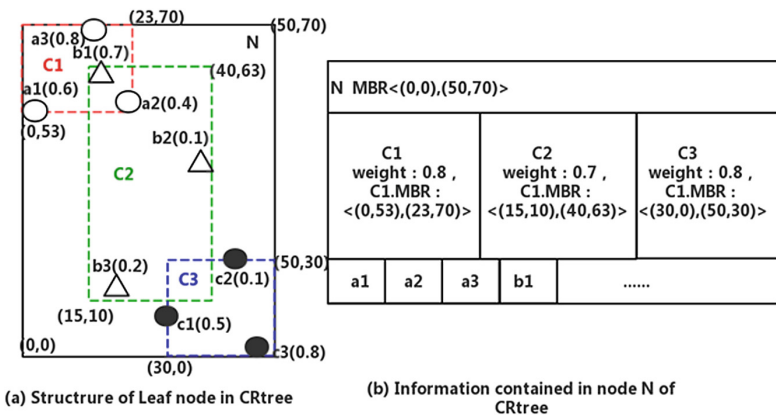


Fig. 2. Structure of a leaf node in the CR-tree

## 4.2 Pruning Principles of CR-Tree

The pruning strategy of CR-tree is based on two theorems.

**Lemma 1.** Let  $p$  be a POI as an entry in the leaf node (or the child node of a non-leaf node)  $N$  of CR-tree, for a user group  $Q$ , we have  $mindist(Q, N) \leq d(Q, p)$ . ■

*Proof.* Whenever  $N$  is a leaf node or a non-leaf node,  $p$  is inside node  $N$ . We can know that  $mindist(q_i, N) \leq dist(q_i, p)$  is true for each  $q_i \in Q$ . Then, for all the users in  $Q$ , we have  $mindist(Q, N) = \sum_{q_i \in Q} mindist(N, q_i) \leq \sum_{q_i \in Q} d(q_i, p) = d(Q, p)$ . Therefore,  $mindist(Q, N)$  is the lower bound of the distance between group-user locations and  $p$ . ■

**Theorem 1.** Let  $N$  be a CR-tree node and  $p$  be a candidate POI inside the tree. In the best case, each  $r_{c_i}^{rad}(p) (c_i \in C_Q^{rad} \wedge c_i \neq c)$  has a maximum value of 1 and the minimum distance between  $p$  and  $p_{c_i}^{rad}(p)$  is 0. Thus, the upper bound of the satisfaction degree of  $N$  over  $Q$ , denoted as  $sd^+(N, Q)$ , is determined by (4).

$$sd^+(N, Q) = \alpha \cdot \left(1 - \frac{mindist(Q, N)}{D_{max} \cdot |Q|}\right) + (1 - \alpha) \cdot \frac{W(c) \cdot r_c^{rad}(N) + |Q| - w(c)}{|Q|} \quad (4)$$

Here,  $r_c^{rad}(N)$  represents the maximum value of comprehensive score of category MBR  $c$  stored in  $N$ . For any POI  $p$  in  $N$ , we have  $sd^+(N, Q) \geq sd(p, Q)$ . ■

*Proof.* According to Lemma 1, we know that  $mindist(Q, N) \leq d(Q, p)$ . Then, we can know:

$$\alpha \cdot \left(1 - \frac{mindist(Q, N)}{D_{max} \cdot |Q|}\right) \geq \alpha \cdot \left(1 - \frac{d(Q, p)}{D_{max} \cdot |Q|}\right) = \alpha \cdot \delta(p, Q)$$

Since  $d(p, p_{c_i}^{rad}(p)) \geq d_{min}(p, p_{c_i}^{rad}(p)) = 0$ , we have:

$$\begin{aligned} (1 - \alpha) \cdot \tau(p, Q) &\leq (1 - \alpha) \cdot \sum_{c_i \in C_Q^{rad}} \frac{1}{|Q|} \cdot \frac{W(c_i) \cdot r_{c_i}^{rad}(p)}{1 + \frac{d_{min}(p, p_{c_i}^{rad}(p))}{rad}} \\ &= (1 - \alpha) \cdot \frac{1}{|Q|} \sum_{c_i \in C_Q^{rad}} W(c_i) \cdot r_{c_i}^{rad}(p) \end{aligned}$$

In addition,  $r_{c_i}^{rad}(p) \leq r_{c_i}^{rad} \max(p) = 1 (c_i \in C_Q^{rad} \wedge c_i \neq c)$ . Since for node  $N$  in  $aR_c, r_c^{rad} \max(p) = r_c^{rad}(N)$ . Thus, we have:

$$\begin{aligned}
 (1 - \alpha) \cdot \tau(p, Q) &\leq (1 - \alpha) \cdot \frac{1}{|Q|} \sum_{c_i \in C'_Q} W(c_i) \cdot r_{c_i}^{rad}(p) \\
 &\leq (1 - \alpha) \cdot \frac{1}{|Q|} \sum_{c_i \in C^{rad}_Q} W(c_i) \cdot r_{c_i}^{rad}_{max}(p) \\
 &= (1 - \alpha) \cdot \frac{W(c) \cdot r_c^{rad}(N) + \sum_{c_i \in C^{rad}_Q \wedge c_i \neq c} W(c_i)}{|Q|}
 \end{aligned}$$

As  $\sum_{c_i \in C'_Q} W(c_i) = |Q|$ , we have  $(1 - \alpha) \cdot \tau(p, Q) \leq (1 - \alpha) \cdot \frac{W(c) \cdot r_c^{rad}(N) + |Q| - W(c)}{|Q|}$ .

Combined the above formulas, we have:

$$\begin{aligned}
 \alpha \cdot \left( 1 - \frac{mindist(Q, N)}{D_{max} \cdot |Q|} \right) + (1 - \alpha) \cdot \frac{w(c) \cdot r_c^{rad}(N) + |Q| - w(c)}{|Q|} \\
 \geq \alpha \cdot \delta(p, Q) + (1 - \alpha) \cdot \tau(p, Q)
 \end{aligned}$$

The left part of the in equation is  $sd^+(N, Q)$  and the right is  $sd(p, Q)$ . Thus, we have proven  $sd^+(N, Q) \geq sd(p, Q)$ . ■

**Theorem 2.** Let  $N.MBR_c$  and  $N'.MBR_{c_i}$  be the corresponding category MBRs in nodes  $N$  and  $N'$  of CR-tree, and  $p$  and  $p'_{c_i}(p)$  be any POIs inside  $N.MBR_c$  and  $N'.MBR_{c_i}$ . In the best case,  $r_c^{rad}(p)$  in CR-tree has a maximum value of the ratings of the category MBR  $c$  in  $N$  and each  $r_{c_i}^{rad}(p)$  has a maximum value of the ratings of the category MBR  $c_i$  in  $N'$ . Suppose that  $mindist(N, N')$  denotes the minimum distance between  $p$  and  $p'_{c_i}(p)$  in  $N'$ . Then, we define the satisfaction degree of  $N$  and  $N'$  over  $Q$  by (5).

$$\begin{aligned}
 sd^+(N, N', Q) &= \alpha \cdot \left( 1 - \frac{mindist(Q, N)}{D_{max} \cdot |Q|} \right) + \\
 &(1 - \alpha) \cdot \frac{w(c) \cdot r_c^{rad}(N.MBR_c) + \sum_{c_i \in C^{rad}_Q \wedge c_i \neq c} w(c_i) \cdot r_{c_i}^{rad}(N'.MBR_{c_i})}{|Q| \cdot \left[ 1 + \frac{mindist(Q, N')}{rad} \right]} \quad (5)
 \end{aligned}$$

Here,  $r_c^{rad}(N.MBR_c)$  and  $r_{c_i}^{rad}(N'.MBR_{c_i})$  represent the max ratings stored in  $N.MBR_c$  and  $N'.MBR_{c_i}$ . For any POI  $p$  in  $N$  and  $p'_{c_i}(p)$  in  $N'$ , the satisfaction degree  $sd(p, Q)$  must satisfy:  $sd^+(N, N', Q) \geq sd(p, Q)$ . ■

*Proof.* According to Lemma 1, we have  $\alpha \cdot \left( 1 - \frac{mindist(Q, N)}{D_{max} \cdot |Q|} \right) \geq \alpha \cdot \delta(p, Q)$ . As the distance between any two points within two rectangle is larger than the minimum distance between two rectangles, we get  $d(p, p'_{c_i}(p)) \geq d_{min}(p, p'_{c_i}(p)) = mindist(N, N')$ . Thus,

$$\begin{aligned}
 (1 - \alpha) \cdot \tau(p, Q) &\leq (1 - \alpha) \cdot \sum_{c_i \in C_Q^{rad}} \frac{1}{|Q|} \cdot \frac{W(c_i) \cdot r_{c_i}^{rad}(p)}{1 + \frac{d_{\min}(p, r_{c_i}^{rad}(p))}{rad}} \\
 &= (1 - \alpha) \cdot \sum_{c_i \in C_Q^{rad}} \frac{1}{|Q|} \cdot \frac{W(c_i) \cdot r_{c_i}^{rad}(p)}{1 + \frac{mindist(N, N')}{rad}}
 \end{aligned}$$

In addition, as  $r_{c_i}^{rad}(p) \leq r_{c_i}^{rad} \max(p) = r_{c_i}^{rad}(N' . MBR_{c_i})$  and  $r_c^{rad} \max(p) = r_c^{rad}(N . MBR_c)$ , we have

$$\begin{aligned}
 (1 - \alpha) \cdot \tau(p, Q) &\leq (1 - \alpha) \cdot \sum_{c_i \in C_Q^{rad}} \frac{1}{|Q|} \cdot \frac{W(c_i) \cdot r_{c_i}^{rad} \max(p)}{1 + \frac{mindist(N, N')}{rad}} \\
 &= (1 - \alpha) \cdot \frac{w(c) \cdot r_c^{rad}(N . MBR_c) + \sum_{c_i \in C_Q^{rad} \wedge c_i \neq c} W(c_i) \cdot r_{c_i}^{rad}(N' . MBR_{c_i})}{|Q| \cdot \left[ 1 + \frac{mindist(N, N')}{rad} \right]}
 \end{aligned}$$

Based on all the above formulas, we have proven  $sd^+(N, N', Q) \geq sd(p, Q)$ . ■

According to **Theorems 1** and **2**, we can design the pruning strategies for CR-tree. The key idea is that if the upper bound of the satisfaction degree of a node is less than the current top- $k$ th highest value, the POIs in the node are not possible to be included in the returned results. Thus, we can skip these POIs and improve the query performance.

### 4.3 CR-Tree-Based Pruning of POI Candidates (OPC)

Based on the theorems in Sect. 4.2, we propose the CR-tree-based pruning algorithm for selecting POI candidates in CR-tree. This algorithm is called *OPC* (Optimized Pruning of POI Candidates), as shown in **Algorithm 2**.

First, we use the root node of the CR-tree as the entry parameter of the OPC query algorithm, and make a top-down recursive call to the tree structure. In this process, the pruning operation based on **Theorem 1** is performed at each node. If the first pruning condition is not met, and the currently traversed node  $N$  is a non-leaf node, its child node is added to the queue  $Q_e$ , which is descended according to (3). We check the queue to get the node recursively until the queue is empty; otherwise, we perform the routine *Score\_computing* as shown in **Algorithm 3** and update the global variables  $H$  and  $kth\_bestvalue$  in time, where  $H$  represents the largest heap of  $k$  candidate POIs with the highest satisfaction value and  $kth\_bestvalue$  represents the current  $k$ th largest satisfaction value.

The routine *Score\_computing* is actually a method for querying the corresponding POI combination of the location semantic category around the candidate POI according to the preference list, and the method simultaneously performs pruning according to **Theorem 2**. **Algorithm 3** shows the *Score\_computing* that is invoked by **Algorithm 2**. Here,  $e$  represents a POI,  $N$  represents the root of a sub-tree in the CR-tree, and  $N_c$  is the root of the CR-tree. In this algorithm, we traverse the CR-tree from the root  $N_c$ , delete non-relevant nodes according to **Theorem 2**, and compute the scores for each leaf node.



**Algorithm 2.** OPC ( $N, Q, c, k$ )

**Input:**  $N$  is the root of a subtree of CR-tree,  $Q$  is the set of querying users,  $c$  is a given category,  $k$  is the number of results, and  $R$  is the root of CR-tree.

**Output:**  $H$ , which is a max heap maintaining the selected  $k$  POIs.

```

1  If  $sd^+(N, Q) > kth\_bestvalue$ 
2  |   If  $N$  is a non-leaf node
3  |   |   stored all the child nodes of  $N$  in  $Q_e$ ;
4  |   |   Pop the first node  $N_s \in Q_e$  and execute OPC( $N_s, Q, c, k$ ) till  $Q_e = \emptyset$ ;
5  |   Else
6  |   |   Get each POI  $e$  labeled with  $c$  and located at  $N$ 
7  |   |    $Score\_computing(e, N, R)$ ;
8  |   |   Update  $H$  and  $kth\_bestvalue$ ;
9  Return  $H$ ;

```

**Algorithm 3.**  $Score\_computing(e, N, N_c)$ 

**Input:**  $N$  is a leaf node of CR-tree,  $N_c$  is node of CR-tree,  $e$  is a given POI located at leaf node of CR-tree

**Output:**  $H$ , which is a max heap maintaining the selected  $k$  POIs.

```

1  If  $sd^+(N, N_c) > kth\_bestvalue$  //Theorem 2
2  |   If  $N_c$  is a non-leaf node
3  |   |   Execute  $Score\_comput$  recursively for each child node of  $N_c$ ;
4  |   Else
5  |   |   Get a set  $S$  of  $e'$  which covers  $c_i \in C_Q^{rad} \wedge c_i \neq c$ ;
6  |   |   If  $dist(e, e') \leq rad$ 
7  |   |   |   Compute  $sd(e, Q)$  and Update  $H$  and  $kth\_bestvalue$ ;
8  Return  $H$ ;

```

**4.4 Further Optimization (CR-Tree\* and the OPC\* Algorithm)**

In real applications, assume that a group of users are looking for a certain restaurant to have dinner together. If they want to visit other kinds of POIs, e.g., theaters, it is very likely for them to look around the POIs within the certain range of the restaurant, such as a range of 1 km around the restaurant. Based on this observation, we propose CR-tree\*, which is an optimized CR-tree that pre-stores the lists of category MBRs in a given range and the corresponding max ratings in different categories.

As shown in Fig. 3, a leaf node of CR-tree is represented by a big rectangle and category MBRs are described by small rectangles inside the node. Some other category MBRs outside the big rectangle belong to other leaf nodes. Then, for category MBR CN, we pre-store the max rating (such as 0.7 for category c1) of every category within range  $r$  of CN in an array. In addition, a set of lists is pre-stored along with the score array. If we need to find POIs labeled with c1, c2 and c3, we just load the pre-

stored information of CN rather than traverse the entire CR-tree. The processing of non-leaf nodes is similar to that of leaf nodes, except that but there are no lists for non-leaf nodes. This optimization can improve the query efficiency compared with the OPC algorithm, and the experimental results in Sect. 5 will demonstrate this.

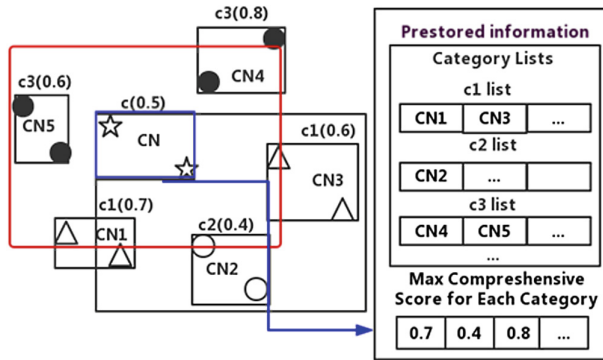


Fig. 3. CR-tree\*: storing additional information in the leaf nodes of CR-tree

## 5 Performance Evaluation

### 5.1 Datasets and Settings

The experimental datasets consist of two real datasets and one simulated dataset. The two real datasets are the POI datasets of Beijing and Guangzhou respectively, including POIs of several cafes, supermarkets, and restaurants. Each record in the dataset contains the latitude and longitude coordinate information of a certain location, and the location category information. In order to indicate users' need for location preference, we randomly assign a float value between 0 and 1 for each POI in the dataset. The properties of the two real datasets are shown in Table 1.

Table 1. Properties of the real datasets

Attribute	Beijing Data	Guangzhou Data
Number of POIs	607307	551595
Number of location categories	14	16
Data size	16.11 MB	14.99 MB
Spatial range	12471 * 10000	13861 * 10000

In addition, we construct a simulation dataset to measure the impact of parameters such as the number of categories, because it is stable in the real datasets. In the simulation dataset, we generate POI sets of sizes 100k, 500k, 1M, 1.5M to 2M. These POI sets are randomly distributed in a two-dimensional space of 10000 \* 10000,

and the corresponding score values are also randomly taken between 0 and 1. There are 8 categories of POIs by default, and each category is the same number.

All algorithms are implemented in Java on a computer with an Intel(R) Core(TM) i3-4210M CPU @2.60 GHz with 8 GB RAM. The index structure is maintained in memory, and the maximum number entries for a node are set to 100. We mainly measure the pruning rate, query time, and index size for all the algorithms or indexes compared. The pruning rate is defined as the ratio of the pruned POIs to all the POIs in the query range. A higher pruning rate is more efficient because it can reduce the search cost. For each query, we run 100 times and use the average response time as the query time.

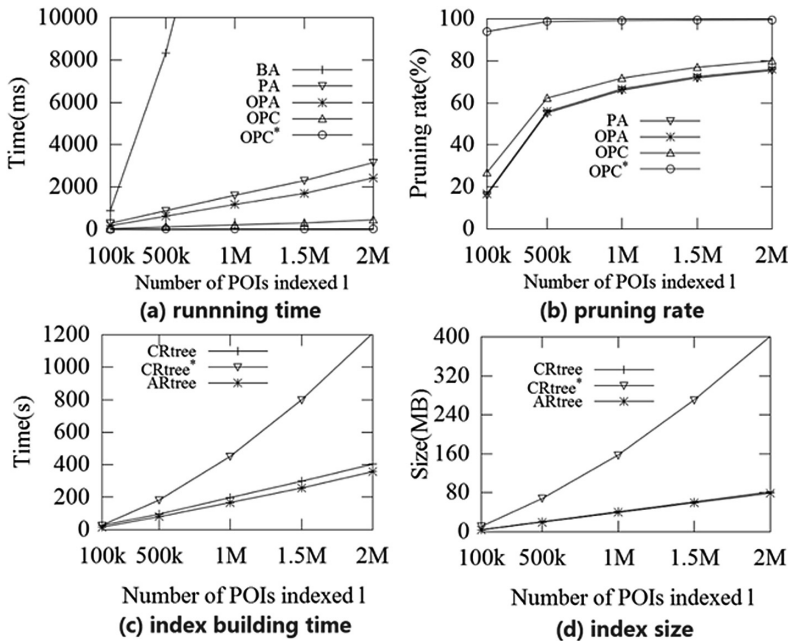


Fig. 4. Results for varying the number of POIs indexed  $l$

### 5.2 Impact of the Number of POIs

In this experiment, we test the impact of the number of POIs on query time, pruning rate, index settling time, and index space cost. It can be seen from Fig. 4(a) that the query time of all algorithms increases gradually with the number of POIs. The BA algorithm is inefficient over large POI datasets. The query time of the OPC algorithm increases with the increase of the number of POIs, but the growth is very slow. Specially, the optimized OPC\* algorithm has relatively stable time performance, because it pre-stores necessary information related to the query so that it does not need to spend time to retrieve the POI set.

Figure 4(b) shows that the pruning rate of all algorithms increases as the number of POIs in the dataset increases. This is mainly because more POIs generated in a plane with a fixed area indicates higher POI density, which has a big impact on the pruning rate. The figure shows that the pruning rate of OPC is higher than that of PA and OPA, which is mainly due to the CR-tree index structure.

Through Fig. 4(a) and (b), we can see that the OPC and OPC\* algorithms are more effective than other methods in terms of pruning rate and query time. Figure 4(c) and (d) show the time when the aR-tree, CR-tree, and CR-tree\* indexes are established and the space occupied by each index. Overall, as the number of indexed POIs increases, the time and space cost for establishing various types of indexes increases. The reason why the growth rate of CR-tree\* is much larger than the other two indexes is that it stores additional information for POIs. As the number of POIs increases, the size needed for pre-stored information increases accordingly. Combined with Fig. 4(a) and (b), it can be concluded that the CR-tree and aR-tree indexes have small gaps in index establishment time and space overhead, but the query efficiency based on the CR-tree index is much higher than that based on the aR-tree index.

### 5.3 Impact of the Number of POI Categories

As the location categories in the real datasets are fixed, we construct a simulated dataset to study the influence of the number of POI categories on the query efficiency. Figure 5 (a) shows that when the category number of POIs changes from 4, 8, 16 to 32, the query time of each query algorithm is gradually reduced. This is because that the number of POIs corresponding to each category is reduced with the increasing of the category number. Figure 5(b) shows that the corresponding query pruning rate increases slowly as the number of POI category increases. We can conclude from Fig. 5 that the OPC and OPC\* algorithms are insensitive to the number of POI category compared with other algorithms.

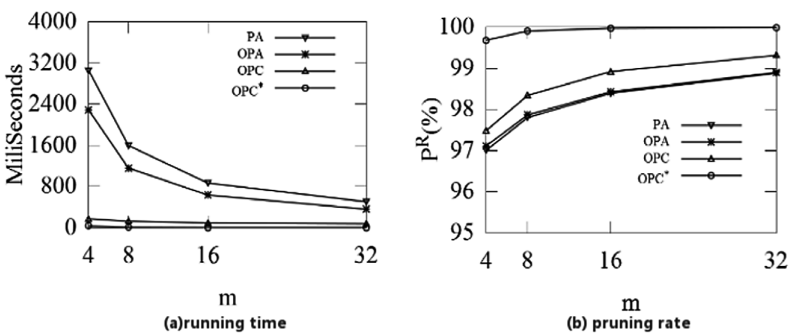


Fig. 5. Results for varying the number of POI categories

### 5.4 Impact of the Number of Preferred Categories in User Groups

Figure 6 shows the change in the query time and pruning rate on the two real datasets as the number of user-preferred query categories increases. Overall, the query time of each algorithm increases as the number of query categories increases and the pruning rate decreases as the number of query categories increases. With the increasing of the number of query categories, the number of POIs that need to be retrieved increases, so does the calculation cost and the query time. However, the OPC\* algorithm is less sensitive to the number of query categories, mainly because its pre-stored information eliminates the need for a large number of traversal index searches for POIs. In addition, different datasets have different POI distributions corresponding to their location categories, so the curve trend of query time and pruning rate varies slightly with query categories.

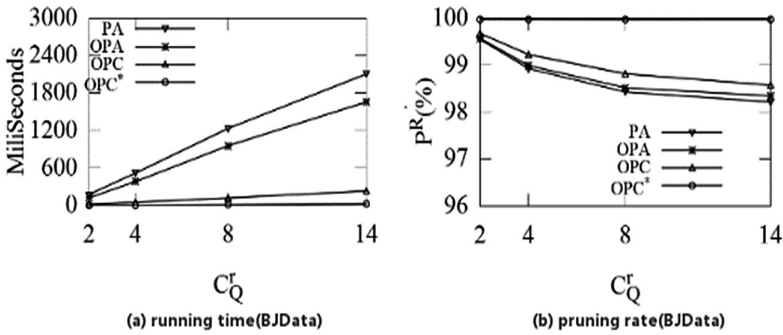


Fig. 6. Results for varying the number of preferred categories in user groups

### 5.5 Impact of Other Parameters

In this section, we report additional experimental results on the impact of other parameters. The Beijing dataset is used as the default dataset in the following experiments.

**Impact of the number of users in group.** The running time increases in PA, OPA and OPC algorithms when the number of users in group, i.e.,  $n$ , increases from 2 to 32 in Fig. 7(a). This is because that more users in group may produce more distance calculation. And a big group may need more preference categories to meet the demands of each one.

**Impact of the range of candidate POI.** To evaluate the effect of range  $r$  of candidate POI, we vary  $r$  from 50 to 200 and the results are shown in Fig. 7(b). It shows that the increase of the range  $r$  mainly leads to the increase of running time for PA, OPA and OPC algorithms. But the running time of OPC increases slowly, comparing with the PA and OPA algorithms. And the OPC\* algorithm is almost not effected by the increasing of  $r$ , which benefits from its pre-stored information.

**Impact of the Number of POIs in the Result.** Figure 7(c) shows that the running time of all algorithms, except OPC\*, increases with the increasing of  $k$ . When  $k$  becomes larger, all algorithms may retrieve more POIs to return the results. This leads to more computations. At the same time, a greater  $k$  value represents the more relaxed constraints of pruning strategies, which is another reason for inefficiency.

**Impact of the Smoothing Parameter.** With the increasing of the smoothing parameter  $\alpha$ , the running time decreases in all algorithms, as shown in Fig. 7(d). The running time of the OPC, OPA and PA algorithms are always in descending order but the OPC\* algorithm shows relatively stable time. When  $\alpha$  is close to 0, the SGP query is similar to a variant top- $k$  spatial preference query. In this case, the algorithm will have low efficiency because of the high cost of complex calculation on  $\tau(p, q)$ . When  $\alpha$  is close to 1, the problem is similar to a group nearest neighbor (GNN) preference query [2].

In general, the OPC\* algorithm is far more efficient than other algorithms and is not sensitive to most parameters. It can be used for small datasets such as datasets within a city range. In addition, we can see that the CR-tree is also efficient for indexing large datasets, such as datasets within a country range.

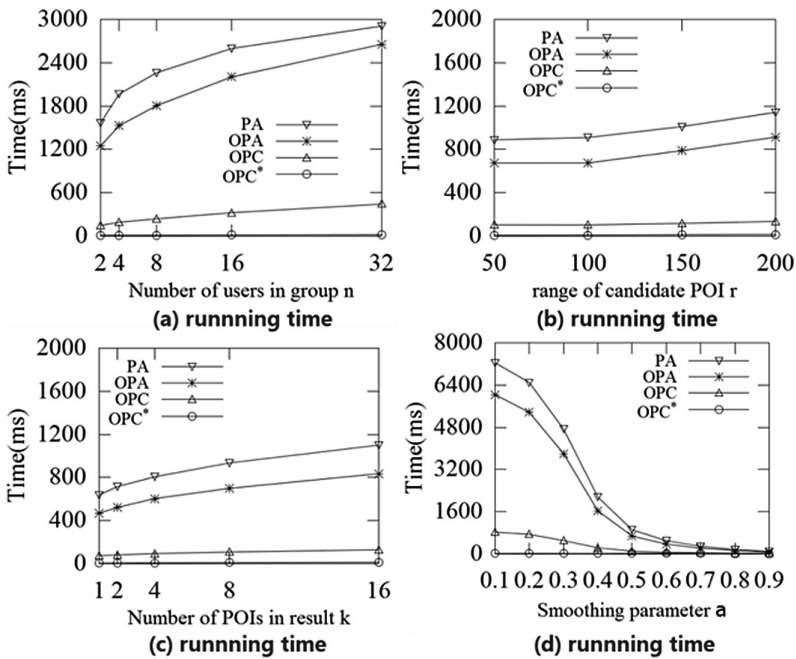


Fig. 7. Results for varying  $n$ ,  $r$ ,  $k$  and  $\alpha$

## 6 Conclusions

In this paper, we propose a new type of spatial queries named SGP queries in social networks. A satisfaction degree model is proposed to measure whether the candidate POI meets the group's needs. Then pruning strategies and a new index structure called CR-tree are proposed to process SGP queries. Experimental results on simulation and real datasets demonstrate the efficiency of our proposal. Our future work will focus on automatically setting the group preference and its weight according to the historical information of each user in the group.

**Acknowledgements.** This work is supported by the National Key Research and Development Program of China (2018YFB0704400, 2018YFB0704404) and the National Science Foundation of China (61672479).

## References

1. Guttman, A.: R-trees: a dynamic index structure for spatial searching. In: SIGMOD, pp. 47–57 (1984)
2. Papadias, D., Shen, Q., Tao, Y., et al.: Group nearest neighbor queries. In: ICDE, pp. 301–312 (2004)
3. Papadias, D., Tao, Y., Mouratidis, K., et al.: Aggregate nearest neighbor queries in spatial databases. *ACM Trans. Database Syst.* **30**(2), 529–576 (2005)
4. Yiu, M.L., Mamoulis, N., Papadias, D.: Aggregate nearest neighbor queries in road networks. *IEEE Trans. Knowl. Data Eng.* **17**(6), 820–833 (2005)
5. Ioup, E., Shaw, K., Sample, J., et al.: Efficient AKNN spatial network queries using the M-tree. In: SIGSPATIAL GIS, p. 46 (2007)
6. Zhu, L., Jing, Y., Sun, W., et al.: Voronoi-based aggregate nearest neighbor query processing in road networks. In: SIGSPATIAL GIS, pp. 518–521 (2010)
7. Li, H., Lu, H., Huang, B., et al.: Two ellipse-based pruning methods for group nearest neighbor queries. In: SIGSPATIAL GIS, pp. 192–199 (2005)
8. Li, F., Yao, B., Kumar, P.: Group enclosing queries. *IEEE Trans. Knowl. Data Eng.* **23**(10), 1526–1540 (2011)
9. Li, Y., Li, F., Yi, K., et al.: Flexible aggregate similarity search. In: SIGMOD, pp. 1009–1020 (2011)
10. Yan, D., Zhao, Z., Ng, W.: Efficient processing of optimal meeting point queries in Euclidean space and road networks. *Knowl. Inf. Syst.* **42**(2), 319–351 (2015)
11. Yiu, M.L., Dai, X., Mamoulis, N., et al.: Top-k spatial preference queries. In: ICDE, pp. 1076–1085 (2007)
12. Gao, Y., Wang, Y., S.: Preference-aware top-k spatio-textual queries. In: Song, S., Tong, Yongxin. (eds.) WAIM 2016. LNCS, vol. 9998, pp. 186–197. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-47121-1\\_16](https://doi.org/10.1007/978-3-319-47121-1_16)
13. Rocha-Junior, J.B., Gkorgkas, O., Jonassen, S., Nørnvåg, K.: Efficient processing of top-k spatial keyword queries. In: Pfoser, D., et al. (eds.) SSTD 2011. LNCS, vol. 6849, pp. 205–222. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-22922-0\\_13](https://doi.org/10.1007/978-3-642-22922-0_13)
14. Cho, H.J., Kwon, S.J., Chung, T.S.: ALPS: an efficient algorithm for top-k spatial preference search in road networks. *Knowl. Inform. Syst.* **42**(3), 599–631 (2015)

15. Attique, M., Cho, H.J., Jin, R., et al.: Top-k spatial preference queries in directed road networks. *ISPRS. J. Geo Infor.* **5**(10), 170 (2016)
16. Rocha-Junior, J.B., Vlachou, A., Doulkeridis, C., Nørnvåg, K.: Efficient processing of top-k spatial preference queries. *Proc. VLDB Endow.* **4**(2), 93–104 (2010)
17. Li, M., Chen, L., Cong, G., et al.: Efficient processing of location-aware group preference queries. In: *CIKM*, pp. 559–568 (2016)
18. Rocha-Junior, J.B., Nørnvåg, K.: Top-k spatial keyword queries on road networks. In: *EDBT*, pp. 168–179 (2012)
19. Tian, Y., Jin, P., Wan, S., Yue, L.: Group preference queries for location-based social networks. In: *APWeb/WAIM*, pp. 556–564 (2017)
20. Zhang, D., Chan, C.-Y., Tan, K.-L.: Nearest group queries. In: *SSDBM*, pp. 7:1–7:12 (2013)





# Reverse-Auction-Based Competitive Order Assignment for Mobile Taxi-Hailing Systems

Hui Zhao<sup>1</sup>, Mingjun Xiao<sup>1(✉)</sup>, Jie Wu<sup>2</sup>, An Liu<sup>3</sup>, and Baoyi An<sup>1</sup>

<sup>1</sup> School of Computer Science and Technology, Suzhou Institute for Advanced Study, University of Science and Technology of China, Hefei, China

xiaomj@ustc.edu.cn

<sup>2</sup> Department of Computer and Information Sciences, Temple University, Philadelphia, USA

<sup>3</sup> School of Computer Science and Technology, Soochow University, Suzhou, China

**Abstract.** Mobile Taxi-Hailing (MTH) is one of the most attractive smartphone applications, through which passengers can reserve taxis ahead for their travels so that the taxi service's efficiency can improve significantly. The taxi-hailing order assignment is an important component of MTH systems. Current MTH order assignment mechanisms fall short in flexibility and personalized pricing, resulting in an unsatisfactory service experience. To address this problem, we introduce a Competitive Order Assignment (COA) framework for the MTH systems. The COA framework mainly consists of the Multi-armed-bandit Automatic Valuation (MAV) mechanism and the Reverse-auction-based Order Assignment (ROA) mechanism. The taxis apply the MAV mechanism to automatically generate the transport service valuations for orders. The platform applies the ROA mechanism to complete each round of order assignment. Then, we analyze the online performance of MAV, and prove that ROA satisfies the properties of truthfulness and individual rationality. Finally, we also demonstrate the significant performances of MAV and ROA through extensive simulations on a real trace.

## 1 Introduction

With the explosive popularity of smartphones, various mobile applications have been developed to make people's lives more convenient. One of the most appealing applications is the Mobile Taxi-Hailing (MTH) system, such as Uber, Didi Chuxing, Lyft, Ola, etc. By using these MTH systems, passengers need not wait for a long time before hailing a taxi, and taxis also will not spend lots of time searching for passengers. Consequently, more and more passengers are willing to use these systems to hail taxis. Statistics show that there have been more than 5 billion Uber trips in 2017 and 15 million daily active riders on average [2].

A typical MTH system consists of a platform residing on the cloud and a collection of passengers and taxi drivers, who have installed the MTH application

in their smartphones, as shown in Fig. 1. If a passenger wants to hail a taxi, he/she would generate a taxi-hailing order and send it to the platform via his/her smartphone. The order includes the start position, destination, and so on. On the other hand, each taxi driver would periodically report its state information to the platform, including the taxi's location, whether the taxi is vacant, etc. After receiving orders from the passengers, the platform would assign each order to a vacant taxi. The order assignment is a vital component of the MTH system.

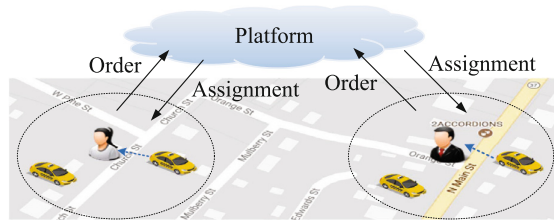


Fig. 1. A typical MTH system

Current MTH systems mainly adopt two types of order assignment strategies. The first is that taxi drivers manually grab the orders publicized by the platform. However, many drivers often complain that they cannot grab any orders most of time, since other drivers might manipulate by using third-party softwares. Another strategy is that the platform directly assigns a vacant taxi to each order according to the distance between them, the reputation of the taxi, etc. However, taxi drivers might be assigned many orders that they do not prefer. In addition, some drivers might wish to compete for their preferred orders by reducing their prices. Nevertheless, this direct assignment strategy has not considered the order competition among taxis and falls short in the personalized pricing requirement. So far, there have been some mechanisms designed for the taxi-hailing order assignment or the taxi dispatch problem, such as [9, 10, 23]. However, although these mechanisms adopt some complex assignment strategies that aim at different optimization objectives, they have still not involved the personalized pricing and competitive order assignment issues.

To enable taxis to flexibly compete for their preferred taxi-hailing orders with personalized prices, we propose a Competitive Order Assignment (COA) framework for MTH systems. The COA framework mainly includes a Multi-armed-bandit Automatic Valuation (MAV) mechanism and a Reverse-auction-based Order Assignment (ROA) mechanism. On the taxi's side, COA allows each taxi to flexibly set valuations for different taxi-hailing orders. First, each taxi uses some pre-defined rules to determine its preferences for different orders, and then determine multiple candidate mark-up pricing strategies as its service charges based on the preferences. For example, an order whose start position is near or destination is located in familiar areas is a preferred order, for which the taxi may ask a lower service charge. Then, once receiving orders from the platform, each taxi can automatically select a strategy to generate its valuation for each order.

Note that different mark-up strategies mean different rewards. Even the same strategy might lead to different rewards in different spatio-temporal scenarios. Thus, the *automatic valuation* is a complex issue. To solve this problem, we see the automatic valuation as a multi-armed bandit process and design the MAV mechanism, by which each taxi can learn to select appropriate strategies maximizing the cumulative rewards. On the platform side, the order assignment is conducted periodically. We treat each round of order assignment as a *reverse auction* process and design the ROA mechanism. In this way, each taxi can rationally compete for its preferred orders via bidding their truthful valuations. More specifically, our major contributions include:

1. We design a competitive order assignment (COA) framework for MTH systems, in which taxis can automatically generate valuations to compete for the taxi-hailing orders.
2. We see the automatic valuation in COA as a multi-armed bandit process, and propose the MAV mechanism. MAV let each taxi automatically select appropriate mark-up pricing strategies for arriving orders to maximize the cumulative rewards. Further, we analyze the online performance of MAV.
3. We model the competitive order assignment in COA as a series of reverse auction processes, and propose the ROA mechanism, including the winner selection and the payment computation. Moreover, we prove that the ROA mechanism is truthful and individually rational.
4. We conduct extensive simulations on a real trace to verify the significant performances of the proposed ROA and MAV mechanisms.

The remainder of the paper is organized as follows. we introduce the COA framework and problem formalization in Sect. 2. The ROA and MAV mechanisms are proposed in Sect. 3. The theoretical analyses are presented in Sect. 4. In Sect. 5, we evaluate the performances of MAV and ROA. After reviewing the related works in Sect. 6, we conclude the paper in Sect. 7.

## 2 Framework and Problem Formalization

### 2.1 The COA Framework

We consider a typical MTH system, including a platform in a cloud, a set of taxis registered in the platform and lots of passengers.

**Definition 1 (Taxi-hailing Order).** A taxi-hailing order is defined as  $o_i = \langle startTime, startLoc, Des \rangle$ , where *startTime*, *startLoc*, and *Des* are the start time, start location, and destination of the corresponding trip, respectively. Moreover, all orders have a common Time-to-Live (TTL). If an order has not been assigned to a taxi during its TTL, it will become invalid.

**Definition 2 (Taxi Driver).** A taxi  $v_j$  is described by its state information:  $s_j = \langle isVacant, startLoc, startTime \rangle$ , where *isVacant* is a boolean indicating whether the taxi is vacant. If *isVacant* is *true*, it means that the taxi can

provide the transport service immediately. Then,  $startLoc$  and  $startTime$  are the current location and time of the taxi, respectively. Otherwise, the taxi is carrying passengers. In this case,  $startLoc$  and  $startTime$  are the destination and arrival time of the current trip, respectively. In addition, to achieve personalized pricing, each taxi  $v_j$  determines its preferences on providing transport services by using some simple pre-defined rules, and then determines multiple candidate mark-up pricing strategies  $\mathcal{K}_j = \{1_j, \dots, K_j\}$  according to the preferences.

The platform continuously receives the taxi-hailing orders from passengers to form an order list  $\mathcal{O}$ , and receives the real-time state information from taxis to form a taxi state list  $\mathcal{S}$ . Then, by comparing the  $startTime$  and  $startLoc$  values of  $s_j \in \mathcal{S}$  and  $o_i \in \mathcal{O}$ , the platform will know whether taxi  $v_j$  can arrive at the start location of order  $o_i$  in time. Therefore, the platform can determine the taxis that can provide the transport service to order  $o_i$ , denoted by  $\mathcal{V}_i$ .

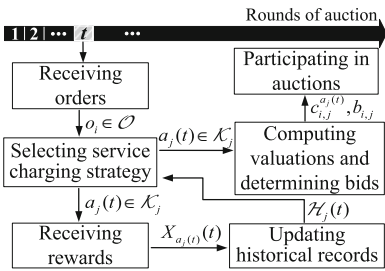


Fig. 2. The automatic valuation process.

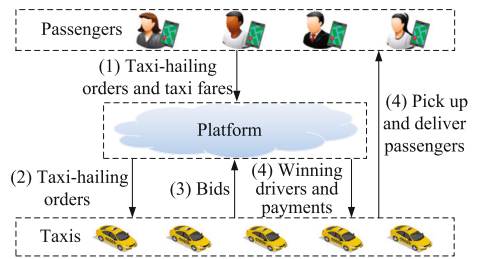


Fig. 3. The reverse auction process.

Based on the above descriptions, we design a Competitive Order Assignment (COA) framework for the MTH system. In the COA framework, taxis set their valuations for arriving taxi-hailing orders based on an automatic valuation mechanism. The platform completes the order assignment via a reverse auction mechanism. The order assignment is conducted periodically, and the period equals to the TTL of orders. At the beginning of each period, the platform conducts a round of assignment for the orders that it has received.

### 2.2 Automatic Valuation Problem Formalization

After receiving taxi-hailing orders from the platform, each taxi selects the mark-up pricing strategy and further generates the valuation for each order automatically. Since different mark-up strategies mean different rewards over different spatio-temporal scenarios, to select appropriate strategies is a complex issue. Therefore, we formalize the automatic valuation as a multi-armed bandit process. A typical multi-armed bandit process consists of a slot machine with multiple arms, each of which is associated with a reward drawn from an initially unknown distribution. A player needs to sequentially select the arms via some

policies, called *bandit policies*, so as to maximize the cumulative reward [3, 13]. Then, the formalization is as follows, which is also illustrated in Fig. 2.

First, we see each taxi  $v_j$  as a player, and its mark-up pricing strategies  $\mathcal{K}_j$  are the arms to be selected. Each arm  $k_j \in \mathcal{K}_j$  is associated with a reward  $X_{k_j}(t)$ . The reward  $X_{k_j}(t)$  is a random variable that is i.i.d. and has unknown probability distribution with a bounded support. Without loss of generality, we assume that  $X_{k_j}(t)$  lies within the range  $[0, 1]$  with a mean  $\mu_{k_j}$ .

Second, the objective of the automatic valuation is to seek a bandit policy to maximize the cumulative reward. Denote the arm selected in the  $t$ -th round as  $a_j(t) \in \mathcal{K}_j$ . The historical records are  $\mathcal{H}_j(t) = \{X_{a_j(1)}(1), \dots, X_{a_j(t)}(t)\}$  with  $\mathcal{H}_j(0) = \emptyset$ . Then, a bandit policy  $\pi_j = (\pi_j(t)_{t=1}^\infty)$  is defined as a sequence of maps  $\pi_j(t) : \mathcal{H}_j(t-1) \rightarrow \mathcal{K}_j$ , which specifies the arm that will be selected under the historical records. Based on this, we define the cumulative reward that taxi  $v_j$  has received up to the  $t$ -th round under the policy  $\pi_j$  as

$$m_j(t) = \sum_{k_j=1}^{K_j} \mu_{k_j} \mathbb{E}[N_{k_j}(t)] \Big|_{\pi_j}. \tag{1}$$

Here,  $N_{k_j}(t)$  is the total number of times that the  $k_j$ -th arm has been selected up to the  $t$ -th round, i.e.,  $N_{k_j}(t) = \sum_{\tau=1}^t \mathbb{1}(a_j(\tau) = k_j)$ , where  $\mathbb{1}(\cdot)$  is an indicator function which is 1 if  $(\cdot)$  is true; otherwise, it equals to 0.

Finally, once selecting an arm  $a_j(t)$ , the valuation, denoted by  $c_{i,j}^{a_j(t)}$ , is determined as the sum of a base price and the mark-up strategy  $a_j(t)$ , in which the base price is the inherent transport cost of the taxi  $v_j$  serving order  $o_i$ . Then, taxis can keep their valuations to participate in the order assignment process. Meanwhile, the taxi will receive a reward  $X_{a_j(t)}(t)$ . Specially, when taxi  $v_j$  loses the order  $o_i$  which the arm  $a_j(t)$ , the reward  $X_{a_j(t)}(t)$  will be 0.

For ease of description, we assume that each taxi deals with one order and selects the arms once in each round. If there are multiple orders for taxi  $v_j$  in a round, we can construct multiple virtual taxis and let each of which deal with one order. In addition, since the valuation for each order is the private information of each taxi, the bandit process of each taxi is also independent of others.

### 2.3 Reverse Auction Problem Formalization

Along with the continuous arrival of taxi-hailing orders, the platform conducts the order assignment based on a reverse auction model. In this model, taxis are seen as the sellers of transport services and the platform holding orders is the buyer. Then, the interactions between passengers and taxis via the platform are described as follows, which is also depicted in Fig. 3.

First, when a passenger wants to start a trip, it submits a taxi-hailing order to the platform. Meanwhile, the passenger will also submit a taxi fare that it is willing to pay for the transport service. Second, the platform receives orders from passengers and publicizes them to taxis. Each order will only be publicized to the taxis that can provide the transport service to it. Third, each taxi receives the orders from the platform. For each received order, taxis determine their true valuations and bids, and then submit all of their bids to the platform.

Here, each bid is not necessarily equal to the corresponding valuation, since each taxi might manipulate the claimed charge. Finally, the platform determines the winners of the auction, computes the payment for each winner, and assigns the corresponding orders.

Consider an arbitrary  $t$ -th round of auction, where the set of orders is  $\mathcal{O}$ , the taxis that can provide the transport service for order  $o_i \in \mathcal{O}$  is  $\mathcal{V}_i$ , and the taxis' valuations and bids are  $\{c_{i,j}^{a_j(t)} | o_i \in \mathcal{O}, v_j \in \mathcal{V}_i\}$  and  $\{b_{i,j} | o_i \in \mathcal{O}, v_j \in \mathcal{V}_i\}$ , respectively. The auction process involves the Winner Selection (WS) problem and the Payment Computation (PC) problem, which are formulated as follows.

First, we consider the optimization objective in each round of order assignment is to maximize the social welfare, defined as follows.

**Definition 3.** The *social welfare* is the total taxi fares of the orders that are assigned to some taxis minus the total valuations of these selected taxis.

Then, we can formalize the WS problem as follows.

**Definition 4.** The *Winner Selection (WS)* problem:

$$\text{Maximize : } \sum_{o_i \in \mathcal{O}, v_j \in \mathcal{V}_i} \phi_{i,j} z_{i,j} \quad (2)$$

$$\text{Subject to : } \sum_{o_i \in \mathcal{O}} z_{i,j} \leq 1, z_{i,j} \in \{0, 1\} \quad (3)$$

$$\sum_{v_j \in \mathcal{V}} z_{i,j} \leq 1, z_{i,j} \in \{0, 1\} \quad (4)$$

**Table 1.** Description of major notations

Variable	Description
$a_j(t)$	the arm that taxi $v_j$ selects in the $t$ -th round of auction
$c_{i,j}^{a_j(t)}, b_{i,j}$	the true valuation and bid of taxi $v_j$ for serving order $o_i$
$r_i$	the taxi fare of the order $o_i$
$N_{k_j}(t)$	the number of times that the $k_j$ -th arm has been selected by taxi $v_j$ up to the $t$ -th round
$X_{k_j}(t), \mu_{k_j}$	the reward that taxi $v_j$ receives from the $k_j$ -th arm in the $t$ -th round, and the mean of its probability distribution
$m_j(t)$	the cumulative reward that taxi $v_j$ receives up to the $t$ -th round
$\hat{\mu}_{k_j}(t)$	the average reward that taxi $v_j$ receives from the $k_j$ -th arm up to the $t$ -th round, i.e., the estimated value of $\mu_{k_j}$
$\mu_j^*$	the mean reward associated with the optimal arm
$\phi_{i,j}$	the value of $r_i - b_{i,j}$

Here,  $z_{i,j} = 1$  indicates that taxi  $v_j$  is selected as the winner to provide the transport service to order  $o_i$ ; otherwise, if  $z_{i,j} = 0$ , order  $o_i$  will not be assigned to taxi  $v_j$ . Moreover, we define  $\phi_{i,j} = r_i - b_{i,j}$ . Note that our reverse auction mechanism is truthful, which means that all taxis will always submit the true valuations as their bids. Hence, we can directly assume  $b_{i,j} = c_{i,j}^{a_j(t)}$ . Then  $\sum_{o_i \in \mathcal{O}, v_j \in \mathcal{V}_i} \phi_{i,j} z_{i,j} = \sum_{o_i \in \mathcal{O}, v_j \in \mathcal{V}_i} (r_i - c_{i,j}^{a_j(t)}) z_{i,j}$  is the social welfare. We will prove the truthfulness in Sect. 4, which implies that this assumption holds.

Finally, the PC problem is defined as follows:

**Definition 5.** The *Payment Computation (PC)* problem is how to determine the payment for each winner so that the whole auction mechanism satisfies the truthfulness and the individual rationality.

**Definition 6 (Truthfulness).** Let  $b_{i,j}$  be an arbitrary bid for taxi  $v_j$  that wins the order  $o_i$ , and  $p_{i,j}(b_{i,j})$  is the corresponding payment determined by the payment computation algorithm of an auction mechanism. Then, if  $p_{i,j}(b_{i,j}) - c_{i,j}^{a_j(t)} \leq p_{i,j}(c_{i,j}^{a_j(t)}) - c_{i,j}^{a_j(t)}$ , we say that the auction mechanism is truthful.

**Definition 7 (Individual Rationality).** For each winning bid  $b_{i,j}$ , the corresponding payoff is nonnegative, i.e.,  $p_{i,j}(b_{i,j}) - c_{i,j}^{a_j(t)} \geq 0$ .

Definition 6 can guarantee that each taxi claims its valuation truthfully, since an untruthful bid will lead to a worse payoff. Definition 7 shows that each taxi can receive a nonnegative payoff if it participates in the auction. In addition, both the reverse auction mechanism and the automatic valuation mechanism need to achieve computational efficiency, defined as follows:

**Definition 8 (Computational Efficiency).** Each round of automatic valuation process and reverse auction process can terminate in a polynomial time.

For ease of reference, we list the main notations of this paper in Table 1.

### 3 The MAV and ROA Mechanisms

#### 3.1 MAV: Automatic Valuation

We have formulated the automatic valuation as a multi-armed bandit process, in which the key is to seek a bandit policy maximizing the cumulative reward of each taxi. Since the distribution of the reward of each arm is unknown a priori, the fundamental challenge in the multi-armed bandit process is to balance the tradeoff between the *exploration* and *exploitation*. On the one hand, taxis have to explore the rewards by randomly selecting the arms. On the other hand, taxis also need to exploit the current knowledge of the rewards to select a best arm.

To solve this bandit dilemma, the Upper Confidence Bound (UCB) policy has been widely used [6, 13]. However, this policy needs to select each arm once a time initially, which is impractical to be applied to our system. The  $\epsilon$ -Greedy policy is another classical policy [20]. This policy selects a random arm with  $\epsilon$ -frequency, and otherwise selects the arm with the current highest estimated expected reward. However, after enough explorations, the estimated rewards will be increasingly close to the true values. The constant factor  $\epsilon \in [0, 1]$  prevents the policy from getting arbitrarily close to the optimal arm. Therefore, in this paper, we first let taxis explore in the first  $t_0$  rounds, where  $t_0 > 0$ . Then, each taxi explores with probability  $t_0/t$ , and exploits with probability  $1 - t_0/t$ . More specifically, in the  $t$ -th round, taxi  $v_j$  selects an arm  $a_j(t) \in \mathcal{K}_j$  according to the

**Algorithm 1.** MAV: Automatic Valuation**Input:**  $v_j, \mathcal{K}_j, t_0 > 0$ **Output:**  $m_j(t)$ 1:  $t \leftarrow 0, m_j(0) \leftarrow 0$ ;2: **if**  $t \leq t_0$  **then**3:  $a_j(t) \leftarrow$  a random arm selected from  $\mathcal{K}$ ;4: **else if**  $\text{rand}() < t_0/t$  **then**5:  $a_j(t) \leftarrow$  a random arm selected from  $\mathcal{K}$ ;6: **else**7:  $a_j(t) \leftarrow \text{argmax}_{k_j \in \mathcal{K}_j} \hat{\mu}_{k_j}(t-1)$ ;8:  $\forall k_j \in \mathcal{K}_j$ , update  $N_{k_j}(t)$  and  $\hat{\mu}_{k_j}(t)$  according to Eq. 8 and Eq. 7, respectively;9:  $m_j(t) \leftarrow m_j(t-1) + X_{a_j(t)}(t)$ ;10:  $t \leftarrow t+1$ ;

following rule: when  $t \leq t_0$ ,  $a_j(t)$  is an arm randomly chosen from the set  $\mathcal{K}_j$ ; when  $t \geq t_0$ ,

$$a_j(t) = \begin{cases} \text{argmax}_{k_j \in \mathcal{K}_j} \hat{\mu}_{k_j}(t-1), & \text{with probability } 1 - \frac{t_0}{t}, \\ \text{a random arm in } \mathcal{K}_j, & \text{with probability } \frac{t_0}{t}. \end{cases} \quad (5)$$

Here,  $\hat{\mu}_{k_j}(t)$  is the estimated value of  $\mu_{k_j}$ . Moreover, we estimate each expected reward value  $\mu_{k_j}$  by averaging the rewards actually received, i.e.,

$$\hat{\mu}_{k_j}(t) = \frac{\sum_{\tau=1}^t X_{k_j}(\tau) \cdot \mathbb{1}(a_j(\tau) = k_j)}{N_{k_j}(t)}, \quad (6)$$

which is equivalent to the following recursive formulas:

$$\hat{\mu}_{k_j}(t) = \begin{cases} \hat{\mu}_{k_j}(t-1), & \text{if } a_j(t) \neq k_j, \\ \frac{\hat{\mu}_{k_j}(t-1) \cdot N_{k_j}(t-1) + X_{k_j}(t)}{N_{k_j}(t)}, & \text{if } a_j(t) = k_j, \end{cases} \quad (7)$$

and,

$$N_{k_j}(t) = \begin{cases} N_{k_j}(t-1), & \text{if } a_j(t) \neq k_j, \\ N_{k_j}(t-1) + 1, & \text{if } a_j(t) = k_j. \end{cases} \quad (8)$$

Based on the above policy, the MAV mechanism automatically selects the mark-up pricing strategies for each taxi in the order assignment process. Then, the taxis can efficiently determine their valuations for each order. The detailed automatic valuation algorithm is shown in Algorithm 1. Since the automatic valuation algorithm is distributed and conducted on each taxi's side, we only display the automatic valuation process of taxi  $v_j$  in Algorithm 1. In Steps 2–7, taxi  $v_j$  selects an arm  $a_j(t)$ . In Steps 8–9, the number of times of each arm that has been selected and the corresponding estimated reward are updated, followed by the computation of cumulative reward.



---

**Algorithm 2.** ROA: Payment Computation (PC)

---

**Input:**  $G = \{\mathcal{O}, \mathcal{V}', \Phi\}, \Psi$

**Output:**  $\{p_{i,j}(b_{i,j}) | \langle o_i, v_j \rangle \in \Psi\}$

- 1: Calculate the total social welfare  $\sum_{\Psi} \phi_{i,j}$  on the matching  $\Psi$ ;
  - 2: **for** each  $\langle o_i, v_j \rangle \in \Psi$  **do**
  - 3:    $\Phi_{-i,j} \leftarrow \Phi - \{\phi_{i,j}\}; G_{-i,j} \leftarrow \{\mathcal{O}, \mathcal{V}', \Phi_{-i,j}\};$
  - 4:   Finding a maximum weighted matching  $\Psi_{-i,j}$  in graph  $G_{-i,j}$ ;
  - 5:   Calculate the total social welfare  $\sum_{\Psi_{-i,j}} \phi_{i,j}$  on the matching  $\Psi_{-i,j}$ ;
  - 6:    $p_{i,j}(b_{i,j}) \leftarrow \sum_{\Psi} \phi_{i,j} - \sum_{\Psi_{-i,j}} \phi_{i,j} + b_{i,j};$
- 

**3.2 ROA: Optimal Winner Selection and Payment Computation**

To select the winners of the auction and determine the order assignment results, we transform the winner selection problem into finding the maximum weighted bipartite matching problem. Consider the  $t$ -th round of order assignment where the set of orders is  $\mathcal{O}$ , the set of taxis is  $\{\mathcal{V}_i | o_i \in \mathcal{O}\}$ , and their bids are  $\{b_{i,j} | o_i \in \mathcal{O}, v_j \in \mathcal{V}_i\}$ . We construct a weighted bipartite graph  $G = \{\mathcal{O}, \mathcal{V}', \Phi\}$ , where  $\mathcal{V}' = \cup_{o_i \in \mathcal{O}} \mathcal{V}_i$  and  $\Phi = \{\phi_{i,j} | o_i \in \mathcal{O}, v_j \in \mathcal{V}_i \subseteq \mathcal{V}'\}$ . Here, the order set  $\mathcal{O}$  and the taxi set  $\mathcal{V}'$  are two separate vertex sets. Set  $\Phi$  indicates the edges across  $\mathcal{O}$  and  $\mathcal{V}'$ , and  $\phi_{i,j} = r_i - b_{i,j}$  is the weight of edge  $\langle o_i, v_j \rangle$ . With the graph  $G$ , we can apply an existing maximum weighted matching algorithm, which has polynomial-time computational complexity, such as the famous Kuhn-Munkres algorithm [12, 16], to get the optimal matching results. Let  $\Psi = \{\langle o_i, v_j \rangle\}$  be the optimal matching with maximum weight in graph  $G$ . Then, we can get the winners and the order assignment results. We set  $z_{i,j} = 1$  if  $\langle o_i, v_j \rangle \in \Psi$ ; otherwise,  $z_{i,j} = 0$ .

In order to ensure that each taxi truthfully reports its true valuation, we compute the payment to each winning taxi based on the VCG auction [17]. The VCG auction can guarantee the truthfulness when the optimal assignment can be achieved. In VCG auction, the winner will be paid with the “externalities” that its presence incurs to others. More specifically, for a given weighted bipartite graph  $G = \{\mathcal{O}, \mathcal{V}', \Phi\}$  and the optimal matching  $\Psi = \{\langle o_i, v_j \rangle\}$ , the payment of a winning bid  $b_{i,j}$  can be determined as follows.

First, we consider a winner selection with the bid  $b_{i,j}$ , and the matching solution is  $\Psi$ . Then,  $\sum_{\Psi} \phi_{i,j} - \phi_{i,j}$  denotes the total social welfare produced by the matching  $\Psi$  except for the single social welfare  $\phi_{i,j}$ . Second, we consider a winner selection without the bid  $b_{i,j}$ . We remove edge  $\langle o_i, v_j \rangle$  from  $G$  to get the corresponding weighted bipartite graph without  $b_{i,j}$ , denoted by  $G_{-i,j}$ , i.e.,  $G_{-i,j} = \{\mathcal{O}, \mathcal{V}', \Phi_{-i,j}\}$ , where  $\Phi_{-i,j} = \Phi - \{\phi_{i,j}\}$ . Then, we conduct the same maximum weighted matching algorithm over  $G_{-i,j}$  to get a matching solution, denoted by  $\Psi_{-i,j}$ . Then,  $\sum_{\Psi_{-i,j}} \phi_{i,j}$  denotes the total social welfare without the presence of bid  $b_{i,j}$ . Finally, the payment  $p_{i,j}(b_{i,j})$  satisfies:

$$r_i - p_{i,j}(b_{i,j}) = \sum_{\Psi_{-i,j}} \phi_{i,j} - (\sum_{\Psi} \phi_{i,j} - \phi_{i,j}), \tag{9}$$

which implies

$$p_{i,j}(b_{i,j}) = \sum_{\Psi} \phi_{i,j} - \sum_{\Psi_{-i,j}} \phi_{i,j} + b_{i,j}. \quad (10)$$

The detailed payment computation algorithm is shown in Algorithm 2. The total social welfare on the matching  $\Psi$  is calculated in Step 1. For each winning bid  $b_{i,j}$ , the weighted bipartite graph  $G_{-i,j}$  is constructed in Step 3. In Steps 4–5, the maximum weighted matching algorithm is conducted over  $G_{-i,j}$  and the total social welfare on the new matching  $\Psi_{-i,j}$  is calculated. Then, the payment  $p_{i,j}(b_{i,j})$  is computed in Step 6.

## 4 Theoretical Analysis

### 4.1 Online Performance of MAV

To analyze the online performance of the MAV mechanism, we derive the expected *regret* of an arbitrary taxi  $v_j$ . First, we consider an Oracle policy, which knows the value of  $\mu_{k_j}$  and can select the optimal arm in each round. Let  $\mu_j^*$  be the mean value associated with the optimal arm, i.e.,  $\mu_j^* = \max_{k_j \in \mathcal{K}_j} \mu_{k_j}$ . Next, we define the loss of selecting the  $k_j$ -th arm as  $\Delta_{k_j} = \mu_j^* - \mu_{k_j}$ . Then, the expected regret, denoted by  $R_j(t)$ , can be defined as the loss in cumulative reward compared with the Oracle policy, i.e.,

$$R_j(t) = \mu_j^* \cdot t - m_j(t) = \mu_j^* \cdot t - \sum_{k_j=1}^{K_j} \mu_{k_j} \mathbb{E}[N_{k_j}(t)] = \sum_{k_j: \mu_{k_j} < \mu_j^*} \mathbb{E}[N_{k_j}(t)] \Delta_{k_j}. \quad (11)$$

Based on this, we can derive the following theorem.

**Theorem 1.** For any  $\rho > 1$ ,  $t \geq t_0$ , the probability that taxi  $v_j$  selects a suboptimal arm  $l_j$  under Algorithm 1 is at most

$$\mathbb{P}[a_j(t) = l_j] \leq \frac{t_0}{tK_j} + \left(1 - \frac{t_0}{tK_j}\right)(\alpha + \beta), \quad (12)$$

where  $\alpha = \frac{4}{\Delta_{l_j}^2} \exp\left(\frac{\Delta_{l_j}^2}{2}\right) \left(\frac{t_0}{t}\right)^{\frac{t_0 \Delta_{l_j}^2}{2\rho K_j}}$ ,  $\beta = \frac{2t_0}{\rho K_j} \left(\frac{t_0}{t}\right)^{\frac{qt_0}{\rho K_j}} \ln\left(\frac{e^2 t}{t_0}\right)$ ,  $q = \frac{3(\rho-1)^2}{8\rho-2}$ .

To prove this theorem, we will make use of the following two inequalities for bounded random variables.

**Lemma 1 (Chernoff-Hoeffding bound).** Suppose that  $X_1, X_2, \dots, X_n$  are  $n$  random variables with common range  $[0, 1]$ , satisfying  $\mathbb{E}[X_t | X_1, \dots, X_{t-1}] = \mu$  for  $\forall t \in [1, n]$ . Let  $S_n = X_1 + \dots + X_n$ . Then, for any  $a \geq 0$ , we have:

$$\mathbb{P}[S_n \geq n\mu + a] \leq \exp(-2a^2/n), \mathbb{P}[S_n \leq n\mu - a] \leq \exp(-2a^2/n).$$

**Lemma 2 (Bernstein inequality).** Suppose that  $X_1, X_2, \dots, X_n$  are  $n$  random variables with common range  $[0, 1]$ , and  $\sum_{t=1}^n \text{Var}[X_t | X_1, \dots, X_{t-1}] = \sigma^2$ . Let  $S_n = X_1 + \dots + X_n$ . Then, for any  $a \geq 0$ , we have:

$$\mathbb{P}[S_n \geq \mathbb{E}[S_n] + a] \leq \exp(-3a^2/(6\sigma^2 + 2a)), \mathbb{P}[S_n \leq \mathbb{E}[S_n] - a] \leq \exp(-3a^2/(6\sigma^2 + 2a)).$$

**Proof:** For some  $\rho > 1$ , let  $x_0 = \frac{1}{\rho K_j} (t_0 + \sum_{\tau=t_0+1}^t \frac{t_0}{\tau})$ . The probability that taxi  $v_j$  selects the  $l_j$ -th arm in the  $t$ -th round is

$$\mathbb{P}[a_j(t) = l_j] \leq \frac{t_0}{tK_j} + (1 - \frac{t_0}{t}) \mathbb{P}[\hat{\mu}_{l_j}(t) \geq \hat{\mu}_j^*(t)], \tag{13}$$

in which

$$\mathbb{P}[\hat{\mu}_{l_j}(t) \geq \hat{\mu}_j^*(t)] \leq \mathbb{P}[\hat{\mu}_{l_j}(t) \geq \mu_{l_j} + \frac{\Delta_{l_j}}{2}] + \mathbb{P}[\hat{\mu}_j^*(t) \leq \mu_j^* - \frac{\Delta_{l_j}}{2}]. \tag{14}$$

The analyses of both the terms in the right hand side of Eq. 14 are the same. Let  $N_{l_j}^{(R)}(t)$  be the number of times that the  $l_j$ -th arm has been selected in the exploration stage up to the  $t$ -th round. Then we have,

$$\begin{aligned} & \mathbb{P}[\hat{\mu}_{l_j}(t) \geq \mu_{l_j} + \frac{\Delta_{l_j}}{2}] = \sum_{\tau=1}^t \mathbb{P}[N_{l_j}(t) = \tau; \hat{\mu}_{l_j}(\tau) \geq \mu_{l_j} + \frac{\Delta_{l_j}}{2}] \\ &= \sum_{\tau=1}^t \mathbb{P}[N_{l_j}(t) = \tau | \hat{\mu}_{l_j}(\tau) \geq \mu_{l_j} + \frac{\Delta_{l_j}}{2}] \cdot \mathbb{P}[\hat{\mu}_{l_j}(\tau) \geq \mu_{l_j} + \frac{\Delta_{l_j}}{2}] \\ &\leq \sum_{\tau=1}^t \mathbb{P}[N_{l_j}(t) = \tau | \hat{\mu}_{l_j}(\tau) \geq \mu_{l_j} + \frac{\Delta_{l_j}}{2}] \cdot \exp(-\frac{\Delta_{l_j}^2 \tau}{2}) \\ &\quad (\text{according to the Chernoff-Hoeffding bound in Lemma 1}) \\ &\leq \sum_{\tau=1}^{\lfloor x_0 \rfloor} \mathbb{P}[N_{l_j}(t) = \tau | \hat{\mu}_{l_j}(\tau) \geq \mu_{l_j} + \frac{\Delta_{l_j}}{2}] + \frac{2}{\Delta_{l_j}^2} \exp(-\frac{\Delta_{l_j}^2 \lfloor x_0 \rfloor}{2}) \\ &\leq x_0 \cdot \mathbb{P}[N_{l_j}^{(R)}(t) \leq x_0] + \frac{2}{\Delta_{l_j}^2} \exp(-\frac{\Delta_{l_j}^2 \lfloor x_0 \rfloor}{2}). \end{aligned} \tag{15}$$

Since  $\mathbb{E}[N_{l_j}^{(R)}(t)] = \frac{t_0}{K_j} + \sum_{\tau=t_0+1}^t \frac{t_0}{\tau K_j} = \rho x_0$ , and  $\text{var}[N_{l_j}^{(R)}(t)] = \sum_{\tau=t_0+1}^t ((\frac{t_0}{K_j} + \frac{t_0}{\tau K_j}) - (\frac{t_0}{K_j} + \frac{t_0}{\tau K_j})^2) \leq \mathbb{E}[N_{l_j}^{(R)}(t)] = \rho x_0$ , according to the Bernstein inequality given in Lemma 2, we have,

$$\mathbb{P}[N_{l_j}^{(R)}(t) \leq x_0] = \mathbb{P}[N_{l_j}^{(R)}(t) \leq \rho x_0 - (\rho - 1)x_0] \leq \exp(-qx_0), \tag{16}$$

where  $q = \frac{3(\rho - 1)^2}{8\rho - 2}$ .

Next, we derive the upper and lower bounds on  $x_0$ . Since  $x_0 = \frac{1}{\rho K_j} (t_0 + \sum_{\tau=t_0+1}^t \frac{t_0}{\tau}) = \frac{t_0}{\rho K_j} (1 + \sum_{\tau=t_0+1}^t \frac{1}{\tau})$ , and  $\ln(\frac{t}{e t_0}) \leq \sum_{\tau=t_0+1}^t \frac{1}{\tau} \leq \ln(\frac{e^2 t}{t_0})$ , we have

$$\frac{t_0}{\rho K_j} \ln(\frac{t}{t_0}) \leq x_0 \leq \frac{t_0}{\rho K_j} \ln(\frac{e^2 t}{t_0}). \tag{17}$$

Combining Eqs. 15–17, we have

$$\mathbb{P}[\hat{\mu}_{l_j}(t) \geq \mu_{l_j} + \frac{\Delta_{l_j}}{2}] \leq \frac{t_0}{\rho K_j} \left(\frac{t_0}{t}\right)^{\frac{qt_0}{\rho K_j}} \ln\left(\frac{e^2 t}{t_0}\right) + \frac{2}{\Delta_{l_j}^2} \left(\frac{t_0}{t}\right)^{\frac{t_0 \Delta_{l_j}^2}{2\rho K_j}} \exp\left(\frac{\Delta_{l_j}^2}{2}\right).$$

In the same way, we can obtain

$$\mathbb{P}\left[\hat{\mu}_j^*(t) \leq \mu_j^* - \frac{\Delta_{l_j}}{2}\right] \leq \frac{t_0}{\rho K_j} \left(\frac{t_0}{t}\right)^{\frac{qt_0}{\rho K_j}} \ln\left(\frac{e^2 t}{t_0}\right) + \frac{2}{\Delta_{l_j}^2} \left(\frac{t_0}{t}\right)^{\frac{t_0 \Delta_{l_j}^2}{2\rho K_j}} \exp\left(\frac{\Delta_{l_j}^2}{2}\right).$$

Therefore, according to Eq. 13, the probability  $\mathbb{P}[a_j(t) = l_j]$  is at most

$$\frac{t_0}{t K_j} + \left(1 - \frac{t_0}{t}\right) \left(\frac{4}{\Delta_{l_j}^2} \left(\frac{t_0}{t}\right)^{\frac{t_0 \Delta_{l_j}^2}{2\rho K_j}} \exp\left(\frac{\Delta_{l_j}^2}{2}\right)\right) + \left(1 - \frac{t_0}{t}\right) \left(\frac{2t_0}{\rho K_j} \left(\frac{t_0}{t}\right)^{\frac{qt_0}{\rho K_j}} \ln\left(\frac{e^2 t}{t_0}\right)\right).$$

The theorem holds.  $\blacksquare$

Finally, based on the above theorem, we obtain the following theorem which bounds the expected regret of our bandit policy.

**Theorem 2.** For any  $\rho > 1$ , given parameter  $t_0$  such that  $t_0 \geq \max\left\{\frac{2\rho K_j}{\Delta_{min_j}^2}, \frac{\rho K_j}{q}\right\}$ ,

where  $\Delta_{min_j} = \min_{l_j: \mu_{l_j} < \mu_j^*} \Delta_{l_j}$  and  $q = \frac{3(\rho-1)^2}{8\rho-2}$ . Then, in each  $t$ -th round of auction where  $t > t_0$ , for an arbitrary taxi  $v_j$ , the expected regret produced by the bandit policy described in Algorithm 1 is at most  $\left(\sum_{l_j: \mu_{l_j} < \mu_j^*} \Delta_{l_j}\right) \frac{t_0}{K_j} \ln t + O\left(\frac{1}{t}\right)$ .

## 4.2 Truthfulness, Individual Rationality and Efficiency

**Theorem 3.** The ROA mechanism satisfies the property of truthfulness.

**Proof:** Suppose that taxi  $v_j$  submits an untruthful bid  $b'_{i,j}$  for order  $o_i$ , i.e.,  $b'_{i,j} \neq c_{i,j}^{a_j(t)}$ . The payment to the bid  $b'_{i,j}$  and order assignment result is denoted as  $z'_{i,j}$ . Denote the order assignment result as  $z_{i,j}$  under the case  $b_{i,j} = c_{i,j}^{a_j(t)}$ . Then, there are two cases: (1)  $z'_{i,j} = z_{i,j}$ ; (2)  $z'_{i,j} \neq z_{i,j}$ .

Case 1 ( $z'_{i,j} = z_{i,j}$ ): If  $z'_{i,j} = z_{i,j} = 0$ , it is obviously that the payoffs under the truthful information and the untruthful information are the same and equal to 0. If  $z'_{i,j} = z_{i,j} = 1$ , then the order  $o_i$  is assigned to taxi  $v_j$  with bid  $b_{i,j}$  or  $b'_{i,j}$ . This means that the presence of bid  $b_{i,j}$  or  $b'_{i,j}$  incurs no effect to other assignment results. Thus, we can obtain that  $p_{i,j}(b'_{i,j}) = \sum_{\Psi} \phi_{i,j} - \sum_{\Psi_{-i,j}} \phi_{i,j} + b'_{i,j} = \sum_{\Psi \setminus \{o_i, v_j\}} \phi_{i,j} - \sum_{\Psi_{-i,j}} \phi_{i,j} + r_i$ . This indicates that the payment is independent of the bid submitted by taxi  $v_j$ . Therefore, in this case,  $p_{i,j}(b'_{i,j}) = p_{i,j}(b_{i,j})$ , and the payoffs are the same as well.

Case 2 ( $z'_{i,j} \neq z_{i,j}$ ): Consider the case  $z'_{i,j} = 0$  and  $z_{i,j} = 1$ . This implies that taxi  $v_j$  loses order  $o_i$  when bidding untruthfully, and its payoff is 0. Therefore, the misreporting leads to the less payoff than bidding truthfully. If  $z'_{i,j} = 1$  and  $z_{i,j} = 0$ , which means that taxi  $v_j$  wins order  $o_i$  with the untruthful bid  $b'_{i,j}$ , then the taxi must claim a lower bid, i.e.,  $b'_{i,j} < b_{i,j}$ , and  $b'_{i,j} \leq p_{i,j}(b'_{i,j})$ . Since taxi  $v_j$  loses order  $o_i$  with bid  $b_{i,j}$ , we have  $p_{i,j}(b_{i,j}) \leq b_{i,j}$ . Consequently, its payoff satisfies:  $p_{i,j}(b'_{i,j}) - c_{i,j}^{a_j(t)} \leq b_{i,j} - c_{i,j}^{a_j(t)} = c_{i,j}^{a_j(t)} - c_{i,j}^{a_j(t)} = 0$ . Thus, in this case, the payoff is negative.

Therefore, each taxi cannot increase its payoffs by manipulating its real valuations, which proves the theorem. ■

**Theorem 4.** The ROA mechanism meets the condition of individual rationality.

**Proof:** If a taxi  $v_j$  does not win the order  $o_i$  with the bid  $b_{i,j}$ , the corresponding payoff will be zero. Otherwise, if taxi  $v_j$  wins the order  $o_i$  with bid  $b_{i,j}$ , the corresponding payoff is  $p_{i,j}(b_{i,j}) - c_{i,j}^{a_j(t)}$ . Here, according to Theorem 3, each bid must be submitted truthfully to achieve the best payoff. Then,  $p_{i,j}(b_{i,j}) = p_{i,j}(c_{i,j}^{a_j(t)})$ , which implies  $p_{i,j}(b_{i,j}) - c_{i,j}^{a_j(t)} = \sum_{\Psi} \phi_{i,j} - \sum_{\Psi - i,j} \phi_{i,j}$ . Since  $\sum_{\Psi} \phi_{i,j}$  is the optimal solution of the winner selection problem and  $\sum_{\Psi - i,j} \phi_{i,j}$  is only a feasible solution where the bid  $b_{i,j}$  is absent, we have  $\sum_{\Psi} \phi_{i,j} - \sum_{\Psi - i,j} \phi_{i,j} \geq 0$ . Hence, the payoff is no less than 0. The theorem holds. ■

Next, we prove the computational efficiency of MAV and ROA.

**Theorem 5.** The ROA mechanism and the MAV mechanism both have a polynomial time computation complexity in one round of order assignment.

**Proof:** The ROA mechanism is composed of the winner selection and the payment computation processes. As described in Sect. 3.2, each round of winner selection can be optimally solved with an existing maximal weighted matching algorithm, whose computation complexity is as most  $O(\max\{|\mathcal{O}|, |\mathcal{V}|\}^3)$ . The payment computation is completed in Algorithm 2, which is dominated by Step 4. Thus, the computation complexity is as most  $O(\max\{|\mathcal{O}|, |\mathcal{V}|\}^3 \cdot \min\{|\mathcal{O}|, |\mathcal{V}|\})$ . Therefore, the ROA mechanism can terminate in a polynomial time. The computation overhead of Algorithm 1 of the MAV mechanism is dominated by Step 7, i.e.,  $O(|\mathcal{K}_j| \ln(|\mathcal{K}_j|))$ . Therefore, the MAV mechanism can terminate in a polynomial time. Therefore, the theorem holds. ■

**Table 2.** Simulation settings

Parameter name	Values
the average number of orders per auction period $ \mathcal{O} $	50, <b>100</b> , 150, 200
the average number of taxis per auction period $ \mathcal{V} $	300, 400, 500, <b>600</b>
parameter $t_0$ in the $\frac{t_0}{t}$ -Greedy policy	<b>100</b> , 1000, 5000
parameter $\epsilon$ in the $\epsilon$ -Greedy policy	0.1, 0.01

## 5 Evaluation

### 5.1 Algorithms in Comparison

In order to evaluate the online performance of the MAV mechanism, we implement an automatic valuation algorithm based on the  $\epsilon$ -Greedy policy for comparison [20]. Given a fixed parameter  $\epsilon$ , the  $\epsilon$ -Greedy policy selects the arm  $k_j = \operatorname{argmax}_{l_j \in \mathcal{K}_j} \hat{\mu}_{l_j}(t-1)$  with probability  $1 - \epsilon$ ; otherwise, the policy selects a random arm with probability  $\epsilon$ .

In order to evaluate the order assignment performance of ROA, i.e., the social welfare performance, we implement three other algorithms for comparison: the Greedy (GRY) algorithm, the Nearest Taxi Selection (NTS) algorithm [9], and the Immediate Selection (IS) algorithm. The GRY algorithm conducts the order assignment under the same bipartite weighted graph as which is constructed in the ROA mechanism. Different from the optimal winner selection algorithm, the GRY algorithm always selects the edge with the largest weight until the taxi set or the order set becomes empty. The taxis in the selected edges are the winners. And the orders in the selected edges are assigned to the corresponding taxis. It is noted that the VCG auction requires that the optimal assignment of the orders must be guaranteed. Consequently, we apply the second price auction to determine the winners' payments in the GRY algorithm. The NTS algorithm selects the nearest taxi for each order. The IS algorithm makes the assignment decision immediately after each order arrives at the platform and each order will be assigned to the taxi that has the maximum single social welfare currently.

## 5.2 Simulation Parameters and Settings

In the evaluation, we use a trace of New York City's taxi trips on January, 2016 [1], which is also used in [23]. This trace consists of about 100,000 completed trip records in 24 h (after discarding some obvious inaccurate records), which is at most 100 trips per minute on average. Each trip record in the trace is composed of the pick-up and drop-off locations (shown as latitude/longitude), the pick-up and drop-off times, the trip distance, and the payment details. From these records, we directly extract the *startTime*, *startLoc*, and *Des* values of each order and taxi. According to these values, we derive each round of orders  $\mathcal{O}$  and taxis  $\mathcal{V}$  through determining the period TTL. For each order  $o_i \in \mathcal{O}$ , we also determine the set of serviceable taxis  $\mathcal{V}_i$ .

Since the trace only contains successful transport records without involving any auction mechanisms, there are no records about the taxis' bids. To evaluate MAV and ROA, we first let taxi fare of each order be equal to the payment value in the trace. Next, we generate each valuation  $c_{i,j}^{a_j(t)}$  as follows. First, we set a base price *basePrice<sub>i</sub>* for each order  $o_i$  as the inherent transport cost and let *basePrice<sub>i</sub>* =  $\frac{r_i}{2}$ . Second, we set the number of service charging strategies of  $K_j$  each taxi as 20, which are set as 20 values randomly chosen from [0, 1], denoted as *Ser<sup>L<sub>j</sub></sup>*, ..., *Ser<sup>K<sub>j</sub></sup>*. Third, the rewards  $\{X_{k_j}(t)|v_j \in \mathcal{V}, k_j \in \mathcal{K}_j\}$  in each round of order assignment are randomly sampled from a truncated Gaussian distribution with mean  $\mu_{k_j} \in [0, 1]$ , standard deviation  $\sigma_{k_j} \in [0, 1]$ , and support [0, 1]. Finally, each taxi  $v_j$  selects its service charging strategies by a policy to obtain the maximal cumulative reward, i.e., the minimal cumulative regret. For ease of description, we call the arm selection policy described in Algorithm 1 the  $\frac{t_0}{t}$ -Greedy policy. Then, if taxi  $v_j$  selects the  $a_j(t)$ -th strategy in the  $t$ -th round of order assignment, its valuation for order  $o_i$  will be  $c_{i,j}^{a_j(t)} = \text{basePrice}_i(1 + \text{Ser}^{a_j(t)})$ , meanwhile taxi  $v_j$  will receive the corresponding reward  $X_{a_j(t)}(t)$ .

In addition, we set different values for the parameter  $t_0$  in our  $\frac{t_0}{t}$ -Greedy policy, and set different values for the parameter  $\epsilon$  in the  $\epsilon$ -Greedy policy for concrete comparison. The detailed parameter settings are listed in Table 2, where the default values are in bold fonts.

### 5.3 Evaluation Metrics and Results

The major metrics in our simulations include the *Online Performance* w.r.t. the MAV mechanism, the *Social Welfare*, *Truthfulness*, *Individual Rationality*, *Overpayment Ratio* and *Time Efficiency* w.r.t. the ROA mechanism. Here, the overpayment is the difference between the total payment to winners and the sum of valuations of each winner. Then, the overpayment ratio is defined as:

$$\lambda = \frac{\sum_{\Psi} p_{i,j}(b_{i,j}) - \sum_{\Psi} c_{i,j}^{a_j(t)}}{\sum_{\Psi} c_{i,j}^{a_j(t)}}. \tag{18}$$

It measures the payments paid by the platform to induce the truthfulness of all taxis. The evaluation results are presented as follows.

**Online Performance of MAV.** To evaluate the online performance of the MAV mechanism, we track two performance metrics: the cumulative regret and the frequency of selecting the optimal arm (denoted as “% optimal arm”). The results are shown in Figs. 4(a)–(b), in which each curve is the average output of 1000 times of repeated simulations. We can find that the regret generated by the  $\frac{t_0}{t}$ -Greedy policy grows logarithmically over time, which is consistent with the theoretical analysis in Theorem 2. Moreover, when  $t_0$  is smaller, the regret grows at a slower speed as shown in Fig. 4(b). This illuminates that the policy has learnt well after a small number of pure exploration. We can also discover that the  $\epsilon=0.1$  policy explores more than the  $\epsilon=0.01$  policy, and it finds the optimal arm earlier. The  $\epsilon=0.01$  policy learns more slowly, but it eventually performs better than the  $\epsilon=0.1$  policy. Nevertheless, an optimally tuned  $\frac{t_0}{t}$ -Greedy policy (e.g.,  $t_0=100$ ) performs almost best among other policies.

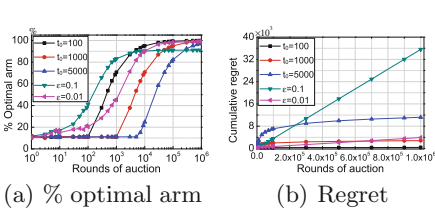


Fig. 4. Evaluation on online performance of MAV.

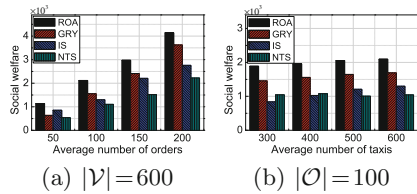
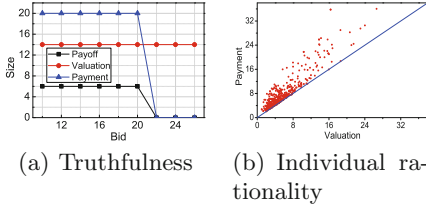
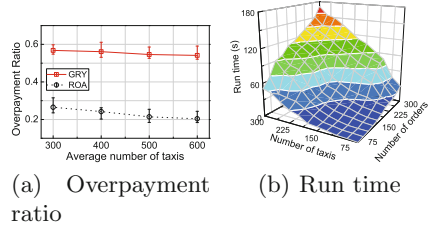


Fig. 5. Evaluation on social welfare of ROA.



**Fig. 6.** Evaluation on truthfulness and individual rationality.



**Fig. 7.** Evaluation on overpayment ratio and time efficiency.

*Social Welfare of ROA.* We evaluate the social welfare performance of the ROA mechanism as follows. First, we set the average number of orders per auction period at  $|\mathcal{O}| = 50, 100, 150,$  and  $200$  by randomly selecting 50 orders in each minute and letting the auction period  $TTL = 1, 2, 3,$  and  $4$  min. Meanwhile, we fix the average number of taxis per auction period at  $|\mathcal{V}| = 600$ . Second, we change  $|\mathcal{V}|$  from 300 to 600, while fixing  $|\mathcal{O}| = 100$ . Finally, we conduct the MAV mechanism with  $t_0 = 100$  to generate the valuations of each taxi. Taxis report their true valuations as their bids to participate in the auction. The results are shown in Figs. 5(a)–(b). On average, the social welfare of ROA is about 79.38%, 61.66% and 52.05% larger than those of GRY, IS and NTS, respectively. Moreover, the social welfare increases with the increasing numbers of orders and taxis, but the increment of the latter is limited. This is because only a few of increased taxis can win the auction.

*Truthfulness and Individual Rationality of ROA.* We verify the truthfulness and individual rationality of ROA under the default settings. First, we randomly select a bid and allow the corresponding taxi to claim a bid different from its real valuation. The result, depicted in Fig. 6(a), shows that the payoff remains unchanged when the taxi's bid is smaller than its valuation. This means that each taxi will still be winner when it claims a lower bid than its current winning bid. However, the payoff is zero when the bid is larger than its payment. This means that the payment paid to each taxi is a critical value ensuring to be a winner. We can hence find that each taxi cannot improve its payoff by bidding untruthfully. To verify the individual rationality, we randomly choose plenty of taxis and orders, and compare the valuation of each taxi with the corresponding payment. The result, plotted in Fig. 6(b), shows that each payment is larger than the corresponding valuation. The individual rationality is also guaranteed.

*Overpayment Ratio and Time Efficiency of ROA.* To evaluate the overpayment ratio performance, we make a comparison with the GRY algorithm. Figure 7(a) shows that the overpayment ratio of ROA is smaller than that of GRY. This implies that the GRY algorithm must pay more so as to induce cooperation from selfish taxis. Moreover, the overpayment ratio of ROA decreases slightly with the increasing number of taxis. This is because that the increasing number of taxis means more taxis with low valuations can be winners, leading to the



reduced overpayment ratio. Second, as shown in Fig. 7(b), when the number of orders is 50, and the number of taxis is 300, the run time is less than 1 min, which is smaller than the auction period. Moreover, when the numbers of orders and taxis are both 300, the run time is no more than 3 min. Therefore, the ROA mechanism can work efficiently in real applications.

## 6 Related Work

In recent years, much attention has been drawn to the study of the taxi-hailing order assignment, taxi dispatch problem, and the task assignment problem in vehicle-based crowdsourcing, such as [9, 10, 14, 15, 18, 21–23]. However, most of these works are based on the direct assignment strategy without involving any auction and personalized pricing mechanisms. Also, many ride sharing services have appeared along with various algorithms on how to match an order to a taxi which can provide the ride sharing service [4, 5, 7, 8], in which the most related works are [4, 5]. Different from our work, [4, 5] do not consider the process in which the taxis' valuations for orders can be learnt or refined over time by observing the historical assignment results. In view of this, we introduce the multi-armed bandit model in our COA framework, by which taxis can automatically price for the orders. Then, they participate in the order auction process. The multi-armed bandit is an online learning model which is widely used in crowdsourcing, cognitive radio networks, etc., [11, 19]. For example, [19] models the unknown expert recruitment problem in crowdsourcing as the multi-armed bandit game, where the unknown experts are seen as arms.

## 7 Conclusion and Future Work

In this paper, we study the order assignment problem in the mobile taxi-hailing systems and propose a competitive order assignment (COA) framework. In COA, we let each taxi automatically set valuations for its preferred orders and design the MAV mechanism. Then, we conduct the competitive order assignment based on a reverse auction and design the ROA mechanism. Further, we analyze the online performance of MAV, and proof that ROA is truthful and individually rational. Moreover, the significance performances of ROA and MAV are also verifies through extensive simulations on a real trace.

**Acknowledgment.** This research was supported in part by National Natural Science Foundation of China (NSFC) (Grant No. 61872330, 61572336, 61572457, 61632016, 61379132, U1709217), NSF grants CNS 1757533, CNS 1629746, CNS 1564128, CNS 1449860, CNS 1461932, CNS 1460971, IIP 1439672, Natural Science Foundation of Jiangsu Province in China (Grant No. BK20131174, BK2009150), Anhui Initiative in Quantum Information Technologies (Grant No. AHY150300), and Natural Science Research Project of Jiangsu Higher Education Institution (No. 18KJA520010).

## References

1. Nyc taxi trips (2016). [http://www.nyc.gov/html/tlc/html/about/trip\\_record\\_data.shtml](http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml)
2. DMR, January 8. <https://expandedramblings.com/index.php/uber-statistics/>
3. Agrawal, R.: Sample mean based index policies by  $o(\log n)$  regret for the multi-armed bandit problem. *Adv. Appl. Probab.* **27**(4), 1054–1078 (1995)
4. Asghari, M., Deng, D., Shahabi, C., Demiryurek, U., Li, Y.: Price-aware real-time ride-sharing at scale: an auction-based approach. In: *ACM SIGSPATIAL* (2016)
5. Asghari, M., Shahabi, C.: An on-line truthful and individually rational pricing mechanism for ride-sharing. In: *ACM SIGSPATIAL* (2017)
6. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.* **47**(2–3), 235–256 (2002)
7. Chen, L., Zhong, Q., Xiao, X., Gao, Y., Jin, P., Jensen, C.S.: Price-and-time-aware dynamic ridesharing. In: *IEEE ICDE* (2018)
8. Chen, L., Gao, Y., Liu, Z., Xiao, X., Jensen, C.S., Zhu, Y.: PTrider: a price-and-time-aware ridesharing system. *Proc. VLDB Endow.* **11**(12), 1938–1941 (2018)
9. Gao, G., Xiao, M., Zhao, Z.: Optimal multi-taxi dispatch for mobile taxi-hailing systems. In: *ICPP* (2016)
10. Garg, N., Ranu, S.: Route recommendations for idle taxi drivers: find me the shortest route to a customer! In: *ACM SIGKDD* (2018)
11. Kang, S., Joo, C.: Low-complexity learning for dynamic spectrum access in multi-user multi-channel networks. In: *IEEE INFOCOM* (2018)
12. Kuhn, H.W.: The Hungarian method for the assignment problem. *Naval Res. Logistics (NRL)* **2**(1–2), 83–97 (1955)
13. Lai, T.L., Robbins, H.: Asymptotically efficient adaptive allocation rules. *Adv. Appl. Math.* **6**(1), 4–22 (1985)
14. Liu, A., et al.: Privacy-preserving task assignment in spatial crowdsourcing. *J. Comput. Sci. Technol.* **32**(5), 905–918 (2017)
15. Liu, A., Wang, W., Shang, S., Li, Q., Zhang, X.: Efficient task assignment in spatial crowdsourcing with worker and task privacy protection. *GeoInformatica* **22**(2), 335–362 (2018)
16. Munkres, J.: Algorithms for the assignment and transportation problems. *J. Soc. Ind. Appl. Math.* **5**(1), 32–38 (1957)
17. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V.V.: *Algorithmic Game Theory*. Cambridge University Press, New York (2007)
18. Tong, Y., Wang, L., Zhou, Z., Chen, L., Du, B., Ye, J.: Dynamic pricing in spatial crowdsourcing: a matching-based approach. In: *ACM SIGMOD* (2018)
19. Tran-Thanh, L., Stein, S., Rogers, A., Jennings, N.R.: Efficient crowdsourcing of unknown experts using bounded multi-armed bandits. *AI* **214**, 89–111 (2014)
20. Vermorel, J., Mohri, M.: Multi-armed bandit algorithms and empirical evaluation. In: Gama, J., Camacho, R., Brazdil, P.B., Jorge, A.M., Torgo, L. (eds.) *ECML 2005. LNCS (LNAI)*, vol. 3720, pp. 437–448. Springer, Heidelberg (2005). [https://doi.org/10.1007/11564096\\_42](https://doi.org/10.1007/11564096_42)
21. Xiao, M., et al.: SRA: secure reverse auction for task assignment in spatial crowdsourcing. In: *IEEE TKDE*, p. 1 (2019)
22. Xiao, M., Wu, J., Huang, L., Cheng, R., Wang, Y.: Online task assignment for crowdsensing in predictable mobile social networks. *IEEE Trans. Mob. Comput.* **16**(8), 2306–2320 (2017)
23. Zheng, H., Wu, J.: Online to offline business: urban taxi dispatching with passenger-driver matching stability. In: *IEEE ICDCS* (2017)



# Top-K Spatio-Topic Query on Social Media Data

Lianming Zhou<sup>1</sup>, Xuanhao Chen<sup>1</sup>, Yan Zhao<sup>2</sup>, and Kai Zheng<sup>1</sup>(✉)

<sup>1</sup> University of Electronic Science and Technology of China, Chengdu, China  
{zhoulianming, chenxuanhao}@std.uestc.edu.cn, zhengkai@uestc.edu.cn

<sup>2</sup> School of Computer Science and Technology, Soochow University, Suzhou, China  
zhaoyan@suda.edu.cn

**Abstract.** With the development of social media and GPS-enabled devices, people can search for what they are interested in more easily. There are many methods, such as spatial keyword query, proposed to help people get useful information. However, most existing methods are based on location and keywords query which neglect the semantic information. In this paper, we propose a new approach named Top-K Spatio-Topic Query (TKSTQ), which takes semantic information into consideration. We use a topic model to obtain topics of texts and organize index based on topic and location. In this way, the query results can satisfy people's requirements better. The experimental results on a real dataset validate that our methods can significantly improve the relevance between result and query.

## 1 Introduction

With the rapid development of social media like Twitter, the scale of User Generated Content (UGC) is increasing. There are about 300 million monthly active users in Twitter, 100 million of which post tweets every day. Thereby, a variety of techniques have been proposed to help users to get useful information from these massive social media data.

In recent years, GPS-enabled mobile devices are widely used so that social media can support location-based service. For example, when a user is posting a tweet, he can choose to tag the location for the tweet. Thus, when something interesting happens, users would like to know what neighboring people are talking about. In addition, new users might want to find some friends with similar hobbies nearby. It is meaningful to find out an approach to recommend useful information to users. Generally, the social media texts (e.g., tweets) are short since the platforms limit the number of characters of a text and short texts are commonly used in daily life, which makes it difficult to get useful information from these texts without effective methods. Therefore, massive work focuses on organizing and analyzing geo-tagged data [1, 6, 9, 13, 15], among which spatial keyword query, a traditional and efficient method, is often adopted to process location-based data.



Fig. 1. Example of query

Figure 1 shows an example of spatial keyword query problem with several users indicated as  $u_1$ ,  $u_2$ ,  $u_3$ ,  $u_4$  and  $u_5$  on the map. A user named *Bob* (i.e., query) likes *Michael Jordan* so much and he wants to find out the people who like *Michael Jordan* too. Therefore he searches *Michael Jordan* on social media. Unfortunately, nobody nearby talks about *Michael Jordan* so that he does not get any useful information. However, there are several tweets posted by  $u_3$  and  $u_4$  talking about *basketball*, which is related to *Michael Jordan*. Obviously,  $u_3$  and  $u_4$  are basketball fans and they are very likely to know *Michael Jordan*, which means that  $u_3$  and  $u_4$  may be the people whom *Bob* is interested in. However, traditional spatial keyword query fails to get them. In this example, *Michael Jordan*, *basketball* and *football* belong to same topic (i.e., sports). Obviously, the similarity between *football* and *Jordan* is not as high as *basketball* and *Jordan*. Hence,  $u_3$  and  $u_4$  satisfy the query better than  $u_1$ . In addition,  $u_3$  is closer to the location of *Bob*, so  $u_3$  will be returned as the result.

Traditional spatial keyword query focuses on locations and keywords, and it performs well in most cases. However, in the above example, spatial keyword query fails to get a satisfactory result because it cannot capture the semantic information, which is the goal of our proposed query. Compared with semantic query, only querying keywords is rigid. Recently, due to the emergence of social media and GPS-enabled mobile devices, the scale of short texts with geo-tagged is getting larger. At the same time, there are many variants of Latent Dirichlet Allocation (LDA) aiming to solve the problem of topic model for short texts, but they do not consider location information, which is important for us to analyze social media data.

In order to improve query effectiveness, we propose a new query named top-k spatio-topic query, which aims to find out topic related objects. We take both location and semantic information of tweets into consideration and design spatio-topic index as well as corresponding query algorithm. To get the topics of tweets, a topic model for short texts is applied to train social media data. Social media texts are generally short and focus on very few topics. In practice, most of

them contain only one topic, thus, we just take one topic from the topic vector. However, traditional spatial keyword index can not deal with geo-tagged data with topics. Therefore, we introduce spatio-topic index combining both location and topic, which consists of two steps. The first step aims to classify all tweets according to the topics and organize them in a quad-tree. In this way, we can query efficiently based on location and topic. In the second step, we design several methods to evaluate the relevance, which includes topic correlation and spatial distance, to rank users. Finally, we conduct extensive experiments on a real data set to demonstrate that our methods are effective.

We make the following contributions in this paper:

- We propose and formulate a novel method, namely Top-K Spatio-Topic Query (TKSTQ). We try to query users on social media according to semantic information and location rather than rigid keywords.
- We apply a topic model in social media data and try to find out the most similar users. Besides, we take both location and topic into consideration and develop a score function to rank users.
- We design spatio-topic index to organize tweets with topics and propose a corresponding efficient query algorithm. In addition, we set threshold to improve the efficiency of query.
- Extensive experiments are conducted with a real-world dataset to evaluate our proposals, in which the empirical results confirm the our solutions are effective.

The remainder of the paper is organized as follows. Section 2 discusses related work and research background. Section 3 formulates the Top-K Spatio-Topic Query (TKSTQ) problem. In Sect. 4, we propose our methodologies, including index organization and query algorithm. In Sect. 5, we show the results of experiments to prove that our methods are efficient. Finally, we conclude our work in Sect. 6.

## 2 Related Work

### 2.1 Social Media

Many efforts have been made to analyze social media and mine information from it. In [17], the authors revealed that the information diffusion follows the Stretched Exponential (SE) model. In [16], the authors took time into consideration and proposed temporal distance metrics. Besides, in [13] and [9], the authors tried to infer location from social media to solve the geo-tag sparseness problem. In [1], the authors discussed the scenarios and uses of location-based services. In [7], the authors proposed a model to analyze the social network and structure. It is worth analyzing geo-tagged data from social media. In this paper, we try to find out top-k spatio-topic results on Twitter.

## 2.2 Spatial Keyword Query

Traditional methods of organizing the geo-tagged data are based on spatial keyword index. In [6], the authors presented a method to find out top-k results by building an IR<sup>2</sup>-tree based on R-tree. In [10], the authors proposed an effective index IR-tree based on R-tree. In [19], the authors proposed preference-based spatial keyword index AIR-tree and combined location and textual relevance to improve queries result. In [15], the authors proposed a scalable spatio-temporal index which can dynamically update with new tweets in. In [18], the authors proposed TOPK-SK and BTOPK-SK based on IL-Quad-tree. In [14], the authors tried to rank spatio-textual objects dynamically and presented STARI index which can search recent results. Besides, in [3, 4], the authors tried to solve some problems in spatial keyword query and proposed new methods to improve the query results.

All methods above are based on spatial keyword query, and in this paper we propose a new method named Top-K Spatio-Topic Query (TKSTQ) which queries according to topic rather than keyword. In this way, users can get results which they really want instead of using the inflexible keywords query to get results.

## 2.3 Topic Model

Topic model is an effective method to obtain semantic information of texts, i.e., topic. The original LDA topic model was proposed in [2]. Since then, a lot of works have proved that LDA works very well. However, for short texts, LDA does not work well because of sparseness of the word co-occurrence so many novel topic models for short texts are proposed recently. In [12], the authors tried to improve topic model with word feature representation. In [5], the authors tried to classify short texts based on LDA and focused on semantic information to reduce the sparseness. In [11], the authors focused on Twitter and proposed Twitter-Network topic model which uses hash tags and followers network. In [22], the authors generated long texts from short texts to improve the effect of topic model. In [8], the authors also extended short texts to long texts, the difference is that biterm is used based on word co-occurrence network. In [20, 21], the authors assumed that short texts focus on a few topics and proposed MetaLDA which improves topic model with document tags and word features. In this paper, we also assume that the number of topics of short texts is small, especially tweet which limits the number of characters.

# 3 Problem Formulation

## 3.1 Tweet Modeling

A tweet is a 4-tuple  $t = (tid, uid, l, W)$ , where  $tid$  is the unique identifier of the tweet,  $uid$  represents the user who posts the tweet,  $l$  is the location where the user posts the tweet and  $W$  is a set of words extracted from the content of the tweet.

Here, we use  $W$  to represent a subset of  $\mathcal{W}$ , which is a vocabulary of all the words. In addition, we use set  $T = \{(tid, uid, l, W)\}$  to denote all tweets in data set,  $U$  to denote all users and  $T_u$  to denote the tweets that user  $u$  posts. Every tweet contains latent information (i.e., topic vector), which is usually a probability distribution. It can be obtained by topic model for short texts.

The amount of social media texts is large but every single one is short. For example, Twitter limits each tweet to 140 characters which can only contain limited information. Besides, there are a lot of stop words in text which are insignificant (e.g., the, this, etc.). Therefore, there are indeed few meaningful words in each tweet so the size of  $t.W$  is always small.

### 3.2 Problem Definition

In the geo-enabled social network, people may want to know *what do people in New York City think about machine learning?* There are so many persons talking about different things. In order to find out suitable results that users want, we formalize it as following problem.

*Problem Definition:* Given a geo-tagged social network data set  $\mathcal{D} = \{T, U\}$ , a set of words  $W$ , a location  $l$  and a number  $k$ . The query  $q = (l, W, k)$  returns a  $k$ -user set  $U_k \subseteq \mathcal{D}.U$  which satisfies the condition:

$$\forall u \in U_k, \forall u' \in \mathcal{D}.U \setminus U_k, score(q, u) \geq score(q, u'),$$

where  $score(\cdot)$  is a function to evaluate the score of users according to distance and topic correlation, and the details will be elaborated in next section.

### 3.3 Score Function

When analyzing the social network data, the most crucial point is semantic information. In this case, topic vector is a good method to represent the semantic information of documents. With the consideration of these aspects, we use topic correlation and distance to measure the score of result.

**Distance.** We define the distance score function as follows.

$$\mu(t, q) = \sigma(r - d(t.l, q.l)), \quad (1)$$

where  $d(t.l, q.l)$  denotes the distance between the location that tweet  $t$  tags with and the location of the query  $q$ ,  $r$  is a constant number which represents the query radius,  $\sigma$  is sigmoid function which is used to normalize distance score. In sigmoid function, if the distance between two locations is less than  $r - 5$  km, the score is close to 1 and the function flattens out. If  $d(t.l, q.l)$  is a large value, the tweet gets very low score and vice versa. The tweets whose  $d(t.l, q.l)$  is smaller than threshold  $r$  are all close to location of the query. In this case, the  $d(t.l, q.l)$  of these tweets has little influence, so we will consider more about semantic score.

On the contrary, if  $d(t.l, q.l)$  is much greater than the threshold  $r$ , their scores are all close to zero because they are useless to the query. Consider the example of Fig. 1, *Bob* would like to find users who are interested in basketball nearby. Here users  $u_1, u_2, u_3$  and  $u_4$  are all close to the location of query. In this case, the sigmoid function can reduce the impact of distance, so that the distance does not affect a lot on final score and the key factor is semantic relevance.

**Topic Correlation.** As mentioned above, all tweets are trained with topic model and each tweet can be represented as a topic vector. The query can be inferred with the model and represented as a vector as well. Therefore, we redefine a tweet as  $t = (tid, uid, l, \mathbf{v})$  and query as  $q = (l, \mathbf{v}, k)$  where  $\mathbf{v}$  denotes topic vector. Based on topic vectors, topic correlation can be calculated with cosine similarity easily, i.e.,

$$\nu(t, q) = \frac{t.\mathbf{v} \cdot q.\mathbf{v}}{\|t.\mathbf{v}\| \times \|q.\mathbf{v}\|}, \quad (2)$$

where  $t.\mathbf{v}$  and  $q.\mathbf{v}$  represent the topic vector of tweet and topic vector of query respectively. Actually topic correlation of any two tweets can be calculated with the equation. Because all the parameters in the topic vector are non-negative, the range of  $\nu(t, q)$  is  $[0, 1]$  as well.

**User Score.** Then we combine distance and topic correlation to define the score of a tweet. Due to the same range of  $\mu(t, q)$  and  $\nu(t, q)$ , they can be combined linearly,

$$\theta(t, q) = \gamma \cdot \mu(t, q) + (1 - \gamma) \cdot \nu(t, q), \quad (3)$$

where  $\gamma$  is a hyper parameter with range of  $[0, 1]$ . User score which is used to rank users is calculated as:

$$score(u, q) = \frac{\sum_{t \in T'_u} \theta(t, q)}{\|T'_u\|}, \quad (4)$$

where  $T'_u$  is a set of tweets of user  $u$  which are filtered by distance and  $\|T'_u\|$  is the number of tweets in the set. In this paper, we calculate the average score of user's tweets. According to the score function, we can find out and sort all users who talk about the relevant topics and are close to the query, then top  $q.k$  users are returned in final result.

## 4 Methodology

### 4.1 Overview

Because spatial keyword query may fail to get results sometimes, we propose top-k spatio-topic query, which can capture the semantic information. Figure 2 shows the architecture of the system. Firstly, external corpus are trained with



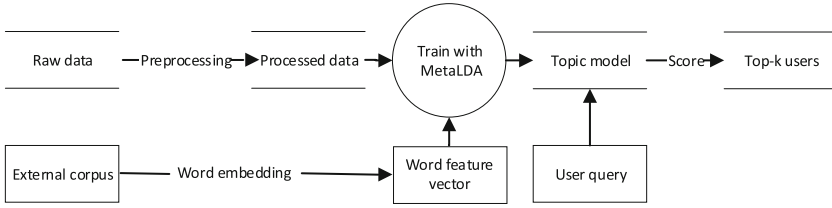


Fig. 2. The architecture of system

word embedding to obtain word feature vectors. Then, the vectors are used as prior knowledge in training of MetaLDA. When a query comes, the texts of query firstly are inferred by topic model to get topics. According to topics and location of query, we construct index therefore we can search for qualified tweets in index and rank corresponding users to get top-k users.

Topic model is an effective way to extract semantic information of documents. In [2], the authors proposed Latent Dirichlet Allocation (LDA) model to obtain the topics of documents and it performs well in most cases. However, for short texts, LDA does not work well due to sparsity of words and mess of content. As mentioned in previous section, tweets are always short so word co-occurrence information is not sufficient for LDA to train the model. Here we use word embeddings to train and obtain word features.

In our system, external corpus such as wikipedia is trained for word embeddings. As mentioned before, word embeddings and preprocessed data are used to train in MetaLDA topic model. Afterwards, each tweet can be represented as a vector which implies the topic distribution.

The training process of MetaLDA topic model is similar to LDA. LDA assumes that each document is a mixture of various topics and each word is also the mixture of some latent topics. Topic distribution has the nature of Dirichlet distribution which is a kind of prior distribution. So the problem is transformed into finding an appropriate distribution to fit data set. The distribution is initialized randomly, then Gibbs sampling is used to train the parameters. After a number of iterations, the distribution converges and topic model is obtained.

After this, spatio-topic index is organized according to the topic and location. When a query is carried out, topics of the query are inferred with MetaLDA and used to get lists of relevant tweets. Then, score function is invoked to calculate score of each user and top-k users are returned.

## 4.2 Index Organization

**Data Organization.** Raw data of tweets consists of a tuple  $(tid, uid, lon, lat, text)$  where they represent tweet id, user id, longitude, latitude, the content of tweet respectively. We split text into a set of words and stop-words are removed at the same time.

But that is not enough in practice because there are still too many words appearing in almost every text while these words are insignificant for extracting

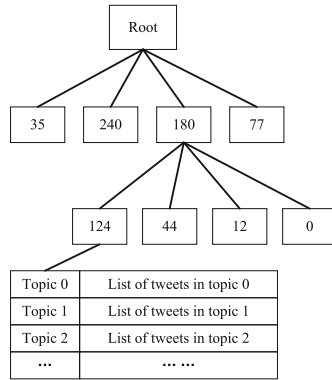


Fig. 3. Structure of index

topics. So we use TF-IDF to remove these words. TF-IDF (i.e., term frequency-inverse document frequency) is a method which can reflect the importance of a word in a collection. Since tweet is short and the set of words is small, term frequency has little effect. So we only use IDF to filter words. We calculate IDF of each word and remove very high and low frequency words to improve the effect of topic model.

After that, the data is trained with MetaLDA using word embedding. Then the topic with the highest probability is chosen to label the tweet. Thus, in the same way, all tweets can be classified according to topic and each topic has a corresponding list of tweets. Meanwhile, the information can be stored in database or distributed file system if the scale of data is large.

---

**Algorithm 1.** Index Construction

---

**Input:** tweets set after training  $T$  and a rectangle area  $A$

**Output:** a quad-tree index

```

1: if  $T.size > M$  and  $Box > R$  then
2:    $node = new\ node(count = T.size, box = A, isleaf = False)$ 
3:   divide  $T.A$  into 4 parts ( $A_1, A_2, A_3, A_4$ )
4:   for  $i = 1; i \leq 4; i++$  do
5:     find all tweets  $T_i$  in  $A_i$ 
6:      $node.child[i] = index\_construction(T_i, A_i)$ 
7:   end for
8: else
9:    $node = new\ node(count = T.size, box = A, isleaf = True, tid\_list = tid\_list)$ 
10: end if
11: return node

```

---

**Spatio-Topic Index.** Quad-tree is used to index location, as shown in Fig. 3. The difference is that the leaf nodes store item lists of topics. They are much

less than the keywords and it benefits efficiency. Threshold is used to avoid too many items in each single node. We divide the whole area into four parts recursively. Lists which contain topics and corresponding tweets are stored in leaves. Algorithm 1 shows the construction of index.

Each node in the quad-tree stores following meta information: *count*, represents the number of tweets in corresponding area, *box*, represents the rectangle area, *isleaf*, is a boolean value which denotes it is leaf or not, *tid\_list*, represents lists of tweets id (if the node is leaf), and *child*, represents four children node.

In order to save memory and improve query efficiency, we set a threshold  $M$  (line 1) to limit the number of tweets in an area. If the size of tweets is larger than  $M$ , we divide the area and call construction recursively (lines 2–7).  $A$  denotes the rectangle area which contains tweets  $T$ . Otherwise the box is a leaf node and stores tweets list of each topic (line 9). Moreover, *tid\_list* keeps lists of tweets id in each topic. It is unavoidable that there might be lots of tweets posted at same place, so we set minimum area  $R$  to ensure the recursion can stop because the rectangle area can not be divided anymore when the area is smaller than  $R$ .

### 4.3 Query Algorithm

We show the detail of the query algorithm TKSTQ in Algorithm 2.

Firstly, we input a query including location  $l$ , words  $W$  and  $k$ . A rectangle box is returned by *get\_range*( $l, r$ ) (line 1). Function *infer*( $W$ ) is invoked to infer the topics of  $W$  with topic model and decides which topics to return (Line 2). In function *infer*( $W$ ), we first infer the topics of  $W$  with topic model, then a topic vector which represents the probability of topics is obtained. Afterwards, we decide which topic to return according to following equation:

$$\lambda_{i-1} = \frac{P_i}{P_{i-1}}, \quad (5)$$

where  $P_i$  denotes the  $i$ -th largest probability in the topic vector. According to Eq. 5, we compute  $\lambda_{i-1}$  in turn. We use a threshold  $\xi$  to decide whether the query belongs to topic  $i$  or not. If  $\lambda_{i-1} > \xi$ , which means that these two topics  $P_i$  and  $P_{i-1}$  have a proximal probability. In this case, we consider that both the topics which  $P_i$  and  $P_{i-1}$  denote are target topics. Besides, we limit the max number of topics to 3 because the query is also a short text which is impossible to contain too many topics.

Assume that there is a query with topic probability 35:0.374,101:0.344, 0.141... . Then we can know that  $P_1 = 0.374$ , and  $P_2 = 0.344$ , thus  $\lambda_1 = 0.344/0.374 = 0.92$ . Suppose that  $\xi = 0.9$ , so the topics for the query are 35 and 101. After that, we search for tweet ids in quad-tree according to the topics and query area to get a list which contains all ids of tweets belonging to topic 35 or 101.

Hereafter we calculate the distance score and semantic score between each tweet and the query. At the same time we will store scores of every user. Finally, we calculate mean score of each user and sort them to return top- $k$  users.

**Algorithm 2.** Query for Top-k Users**Input:** a set of words  $W$ , a location  $l$ , and a number of users to return  $k$ .**Output:** a list of  $k$  users  $(u_1, u_2, \dots, u_k)$ 


---

```

1:  $search\_range = get\_range(l, r)$ 
2:  $query\_topics = infer(W)$ 
3:  $tweet\_list = get\_list(root, search\_range, query\_topics)$ 
4: List users
5: for  $tid$  in  $tweet\_list$  do
6:    $t = get\_tweet(tid)$ 
7:   compute tweet score according to equation 1, 2, 3
8:   if  $t.uid \notin users$  then
9:      $users.append(t.uid, list(tweet\_score))$ 
10:  else
11:     $user.scores.append(tweet\_score)$ 
12:  end if
13: end for
14: for  $usr$  in  $users$  do
15:    $usr.score = mean(tweet\_scores)$ 
16: end for
17:  $users.sort()$ 
18: return top-k users

```

---

## 5 Experiment

In this section, we will show the effectiveness of the proposed method top-k spatio-topic query. We conduct experiments on real data set which includes about one million tweets with geo-tag. We crawl these geo-tagged tweets through *Twitter REST API* and preprocess the data as mentioned above and remove tweets including less than two non-stop words. For those without geo-tag, there are some methods such as [13] and [9] to infer their locations. Here we just focus on geo-tagged tweets. We set  $\gamma = 0.5$ ,  $\xi = 0.9$  and the upper bound of the number of tweets in a box  $M = 1000$ .

### 5.1 Baseline

In our experiments, The advantages of our method can be presented by spatial keyword index and spatio-topic index based on LDA. For the sake of fairness, we implement all methods in the same way and same environment.

- **Spatial keyword index based on keyword** is a classic method in spatial database. For spatial index, there are various indexes like R-tree, IR-tree, etc and here same quad-tree is used to ensure fairness.
- **Spatio-topic index based on LDA.** LDA is an efficient topic model in natural language processing. Similarity, quad-tree and topic index are employed.

Both MetaLDA and LDA are trained with same hyper-parameters, where  $\alpha$  is 0.1, the number of iterations is 2000,  $\beta$  is 0.01 for LDA and  $\beta$  is computed with

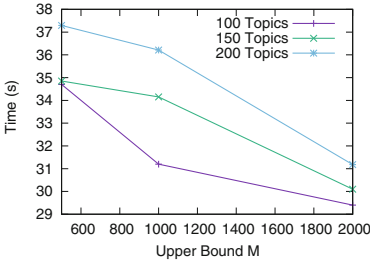


Fig. 4. Construction time

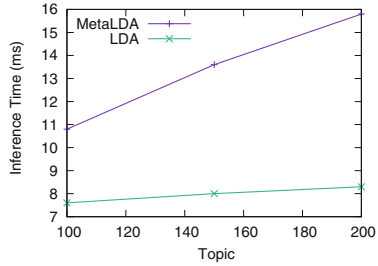


Fig. 5. Inference time

pre-trained word features from wikipedia for MetaLDA. In our experiments, we train MetaLDA and LDA with 100, 150, 200 topics in advance.

### 5.2 Index Construction

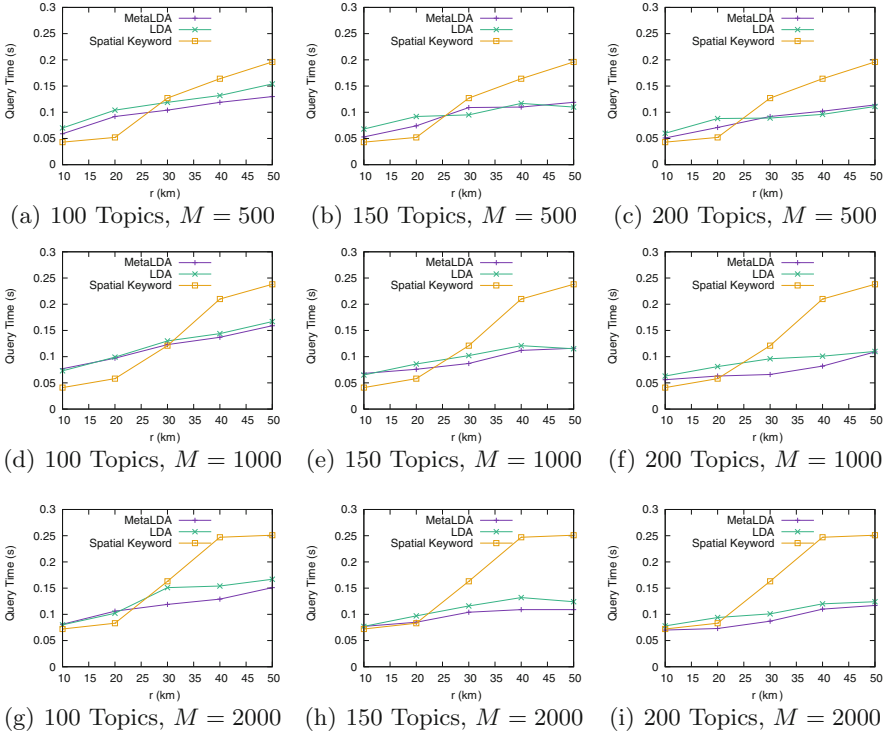
At first, we evaluate the efficient of the index construction and the memory cost. As shown in Table 1, for spatio-topic index, MetaLDA and LDA have no difference in the time of index construction. So the construction time is mainly influenced by the topics. With the number of topics increasing, the time of index construction grows. The time of construction of spatial keyword index is much larger because of the number of keywords is large. Likewise, large scale keywords also lead to large memory cost.

As for memory cost, MetaLDA and LDA have almost same size of index which contains all ids of tweets. The size of the index depends only on the size of the dataset.

Table 1. Index construction

Methods	Topics	Time (s)	Size (MB)
MetaLDA	100	31.20	62.54
	150	34.15	63.50
	200	36.21	64.26
LDA	100	32.35	62.56
	150	33.01	63.48
	200	38.73	64.20
Spatial keyword	/	274.46	167.73

Then we evaluate the effect of  $M$ . Because MetaLDA and LDA have no difference in this respect, we just show MetaLDA. As shown in Fig. 4, the larger upper bound  $M$ , the shorter the construction time. The reason is that larger  $M$  means less nodes in quad-tree so construction costs less time. For topics, 200



**Fig. 6.** Query efficiency with fixed  $M$  500, 1000, 2000

topics need more time because more topics need more lists in each node (see Fig. 3). From the figure, we can see that  $M$  affects the construction time a lot. Moreover, it is clear that it will affect query efficiency too.

### 5.3 Query

To evaluate the efficiency of queries, we randomly select 100 tweets to construct queries. According to the scale of the data set, we set  $k = 5$  and  $r = 10$  km to 50 km. For spatial keyword index, although the number of keywords influences the query efficiency and result, we ignore it here and just use same queries for all methods and parameters. Each experiment repeats 5 times to take the average.

**Query Efficiency.** For MetaLDA and LDA, the situation is different because raw queries can not be used to conduct query in spatio-topic index. They need to be transferred into corresponding format and inferred with topic model, which means that the query time of spatio-topic index is not only related to the index itself, but also related to the inference time. Hence, the total query time includes two parts, inference time and query time in index. As shown in Fig. 5, MetaLDA

costs more time in inference time because it needs more calculation with extra information such as word embedding. While the inference does not need much time (several milliseconds) so it is only a small part of total query time.

Through inference of topic model, all queries can be transformed into a few topics and a location. It is efficient because the influence of the number of the query words is small. We do not need to worry about that there are no match results due to the small number of query words because no matter how many words in query, the text of query always belongs to certain topics.

Figure 6 shows the results of the average query time. Note that in each group of experiments we use same upper bound  $M$  to limit the number of tweets in a rectangle area. As shown in Fig. 6(a), when the number of topics is fixed as 100 and  $M$  fixed as 500, the time two methods (i.e., MetaLDA and LDA) cost is close while spatial keyword index is less efficient. With the number of topics increases, the query time of spatio-topic index is much shorter, especially when the  $r$  is large. For spatio-topic index based on MetaLDA and LDA, there is almost no difference between them. Only difference between them is that topic vectors are different but its impact on the effect is small so that their query time is almost the same.

From Fig. 6 we can see that the number of topics affects query efficiency a lot. Fewer topics mean that the classification is vague and there are more tweets of per topic on average. For example, sports, if the number of topics is 5, then we can divide the sport into basketball, football and so on. But if there is just 1 topic, we can only take sports as topic. However, it does not mean that more topics can always improve the efficiency. If there are too many topics, the semantics between the topics will contain a lot of overlap which could cause chaos. Thus, choosing a suitable number of topics can improve efficiency, usually we decide number of topics from experience and training. As we can see in Fig. 6(d, f), when the number of topics is 100, the query costs more time and performs worse than 200 topics especially when  $r$  is large.

For the upper bound  $M$ , we fix it as 500, 1000 and 2000. When  $M$  is fixed as 2000 and  $r$  is small such as 10 km, as shown in Fig. 6(g), query time is only a little longer compared with  $M = 500$  (see Fig. 6(a)). If  $r$  is small but  $M$  is large, it means that the query might cover larger area and more tweets need to be processed. With the increase of  $r$ , the impact of  $M$  becomes smaller. As mentioned in previous section,  $M$  could affect the index construction and query efficiency.

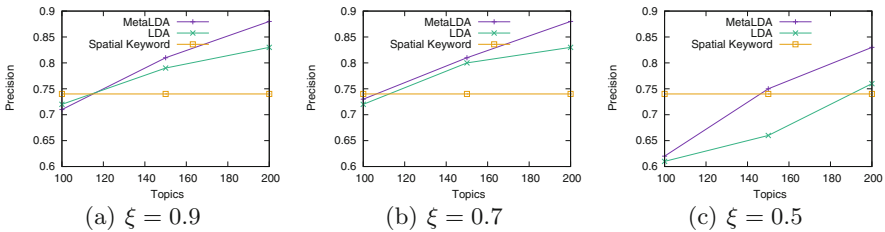


Fig. 7. Precision of results

**Query Precision.** Evaluating the query precision, mainly is to evaluate the similarity between tweets of top-k users and query words. Most traditional evaluation methods only take the number of keywords into consideration but do not fit our original purpose, i.e., trying to query according to semantic information rather than keywords. In order to make our results more convincing. We invite some students to estimate the results. In experiments, we select 22 meaningful texts of tweets to query. For each method we propose, we choose top-5 results in any of the 22 queries, we set  $r = 10$  km. Clearly, for each method, we can get  $5 * 22$  results, we show all results to the students, and they decide whether the result is related to the corresponding query or not. Finally, we take average precision of each method as results.

**Table 2.** Similar content

Query words	Result words
retail	customer, service, store
semester, exam	revised, professor, paper
music	band, chorus, concert, dance
police	officer, gunpoint, shooting
bitcoin	btc, eth

Compared with spatial keyword index, the advantage of our method is that different related words can be found in the result. Our experiment proves that our method is effective because many related words that do not appear in query appear in result. As shown in Table 2, the words related to the keywords of the query appear in the result. For example, if a user searches *music*, however, there may be no tweets nearby which mention *music* directly, but some tweets which talk about *band*, *concert* and so on may relate to it. So these tweets can be returned as results.

In our data set, spatio-topic index with 100 topics does not perform well compared with spatial keyword index because of the scale of the data. If the scale of the data set is smaller, the index with 100 topics might work better. Fewer topics mean that lots of unrelated words might be divided into same topics, which would reduce the precision of results. The index achieves the best effect in this data set when the number of topics is around 200.

In addition,  $\xi$  can also affect the precision. Figure 7 shows the relation between the precision and  $\xi$ . MetaLDA, which is trained with external information (e.g., word embedding), gets higher precision than LDA and spatial keyword query. Besides, when  $\xi$  decreases, the precision of both spatio-topic queries decreases too.

For parameter  $\xi$ , it can decide what topics the query belongs to. Larger  $\xi$  means that the query is more strict with topic, so some results that are less related to the query will be neglected. For example, the query contains *semester*



and *exam*, when  $\xi = 0.5$ , not only these two words but also other words such as *test* and *question* appear in result, while when  $\xi = 0.9$ , the words *test*, *question* disappear in result. From Fig. 7, we can see it clearly that when  $\xi = 0.9$ , the precision is higher (i.e. the result is more related to the query). While it does not mean that higher  $\xi$  is always better, if we want to get a larger range of results, smaller  $\xi$  might be better.

In summary, top-k spatio-topic query is better in efficiency and precision on Twitter data set. In addition, the experiments show that smaller  $M$  and larger number of topics can improve query efficiency. In terms of precision, spatio-topic query can effectively search for semantic related results.

## 6 Conclusion

In this paper, we develop a top-k spatio-topic index and corresponding query (TKSTQ) algorithm on social media data, which can find out top-k users nearby who talk about the relevance information. Different from traditional methods, our method considering both semantic information and location into consideration, utilizes MetaLDA to capture topics of texts, and organizes them efficiently in index. Furthermore, we define a score function, which includes topic correlation and distance score, to rank users, and design an efficient query algorithm. In addition, our method can be used to recommend friends to users according to the tweets. Our experiments on a real Twitter data set demonstrate the effectiveness of our methods.

Our methods have some limitations. The training of topic model costs much time, especially in large scale data. Besides, sometimes topic model may not work very well in short texts. However, to some extent, we can use large scale data to improve the effect of topic model because more words co-occurrence information can be captured.

**Acknowledgement.** This work is supported by the Natural Science Foundation of China (Grant No. 61532018, 61836007, 61832017).

## References

1. Bao, J., Lian, D., Zhang, F., Yuan, N.J.: Geo-social media data analytic for user modeling and location-based services. *SIGSPATIAL Spec.* **3**, 11–18 (2016)
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. *J. Mach. Learn. Res.* **3**, 993–1022 (2003)
3. Chen, L., Lin, X., Hu, H., Jensen, C.S., Xu, J.: Answering why-not questions on spatial keyword top-k queries. In: 2015 IEEE 31st International Conference on Data Engineering, pp. 279–290 (2015)
4. Chen, L., Xu, J., Lin, X., Jensen, C.S., Hu, H.: Answering why-not spatial keyword top-k queries via keyword adaption. In: 2016 IEEE 32nd International Conference on Data Engineering (ICDE), pp. 697–708 (2016)
5. Chen, Q., Yao, L., Yang, J.: Short text classification based on LDA topic model. In: 2016 International Conference on Audio, Language and Image Processing (ICALIP), pp. 749–753 (2016)

6. Felipe, I.D., Hristidis, V., Rishe, N.: Keyword search on spatial databases. In: 2008 IEEE 24th International Conference on Data Engineering, pp. 656–665 (2008)
7. Iijima, R., Kamada, Y.: Social distance and network structures. *Theor. Econ.* **2**, 655–689 (2017)
8. Jiang, L., Lu, H., Xu, M., Wang, C.: Biterm pseudo document topic model for short text. In: 2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI), pp. 865–872 (2016)
9. Lee, K., Ganti, R.K., Srivatsa, M., Liu, L.: When Twitter meets foursquare: Tweet location prediction using foursquare. In: Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, pp. 198–207 (2014)
10. Li, Z., Lee, K.C.K., Zheng, B., Lee, W.C., Lee, D., Wang, X.: IR-tree: an efficient index for geographic document search. *IEEE Trans. Knowl. Data Eng.* **4**, 585–599 (2011)
11. Lim, K.W., Chen, C., Buntine, W.L.: Twitter-network topic model: a full Bayesian treatment for social network and text modeling. *CoRR* (2016)
12. Nguyen, D.Q., Billingsley, R., Du, L., Johnson, M.: Improving topic models with latent feature word representations. *Trans. Assoc. Comput. Linguist.* **3**, 299–313 (2015)
13. Pontes, T., et al.: Beware of what you share: inferring home location in social networks. In: 2012 IEEE 12th International Conference on Data Mining Workshops, pp. 571–578 (2012)
14. Ray, S., Nickerson, B.G.: Dynamically ranked top-k spatial keyword search. In: Proceedings of the Third International ACM SIGMOD Workshop on Managing and Mining Enriched Geo-Spatial Data, pp. 6:1–6:6 (2016)
15. Skovsgaard, A., Sidlauskas, D., Jensen, C.S.: Scalable top-k spatio-temporal term querying. In: 2014 IEEE 30th International Conference on Data Engineering, pp. 148–159 (2014)
16. Tang, J., Musolesi, M., Mascolo, C., Latora, V.: Temporal distance metrics for social network analysis. In: Proceedings of the 2nd ACM Workshop on Online Social Networks, pp. 31–36 (2009)
17. Wang, D., Li, Z., Salamatian, K., Xie, G.: The pattern of information diffusion in microblog. In: Proceedings of the ACM CoNEXT Student Workshop, pp. 3:1–3:2 (2011)
18. Zhang, C., Zhang, Y., Zhang, W., Lin, X.: Inverted linear quadtree: efficient top k spatial keyword search. *IEEE Trans. Knowl. Data Eng.* **7**, 1706–1721 (2016)
19. Zhang, J., Liu, D., Meng, X.: Preference-based top-k spatial keyword queries. In: Proceedings of the 1st International Workshop on Mobile Location-based Service, pp. 31–40 (2011)
20. Zhao, H., Du, L., Buntine, W.: A word embeddings informed focused topic model. In: Zhang, M.L., Noh, Y.K. (eds.) Proceedings of the Ninth Asian Conference on Machine Learning, pp. 423–438 (2017)
21. Zhao, H., Du, L., Buntine, W.L., Liu, G.: MetaLDA: a topic model that efficiently incorporates meta information. *CoRR* (2017)
22. Zuo, Y., et al.: Topic modeling of short texts: a pseudo-document view. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 2105–2114 (2016)

# **Spatio-Temporal**



# A Frequency-Aware Spatio-Temporal Network for Traffic Flow Prediction

Shunfeng Peng, Yanyan Shen<sup>(✉)</sup>, Yanmin Zhu, and Yuting Chen

Department of Computer Science and Engineering,  
Shanghai Jiao Tong University, Shanghai, China  
{dwkegu, shenyy, yzhu, chenyt}@sjtu.edu.cn

**Abstract.** Predicting traffic flow is crucial for transportation management and resource allocation, which has attracted more and more attention from researchers. The traffic flow in a city generally changes over time periods but always exhibits certain periodicity. Previous works focused on modeling spatial and temporal correlations using convolutional and recurrent neural networks respectively. Typically, a method that can effectively absorb more time-interval inputs and integrate more periodic information will achieve better performance. In this paper, we propose a Frequency-aware Spatio-temporal Network (FASTNet) for traffic flow prediction. In addition to modeling the spatio-temporal correlations, we dynamically filter the inputs to explicitly incorporate frequency information for traffic prediction. By applying Discrete Fourier Transform (DFT) on traffic flow, we obtain the spectrum of traffic flow sequence which reflects certain travel patterns of passengers. We then adopt a frequency-based filtering mechanism to filter the traffic flow series based on the explored spectrum information. To utilize the filtered tensor, a 3D convolutional network is designed to extract the spatio-temporal features automatically. Inspired by the frequency spectrum of traffic flows, this spatio-temporal convolutional network has various kernels with different sizes on temporal dimension, which models the temporal correlations with multi-scale frequencies. The final prediction layer summarizes the spatio-temporal features extracted by the spatio-temporal convolutional network. Our model outperforms the state-of-the-art methods through extensive experiments on three real datasets for citywide traffic flow prediction.

**Keywords:** Flow prediction · Filtering mechanism · Frequency spectrum analysis · Spatio-temporal correlation · Convolution

## 1 Introduction

Traffic flow prediction is inevitably important for various transportation services such as route planning and resource allocation [21, 24]. According to the recent

report<sup>1</sup> from Didi, the world's leading mobile transportation platform, there are more than 30 million taxi orders per month, with an average waiting time of more than 7 min. Accurately predicting traffic flow in the near future is beneficial for setting the tidal lanes to decrease congestion timely and rescheduling the vehicles to achieve supply-demand balance. In many real-life applications, the goal of traffic flow prediction is to provide short-term traffic flow-in and flow-out information for city-wide regions, and this goal is achieved by learning spatio-temporal regularities from massive traffic flow statistics in the past, and applying the insights for future flow prediction.

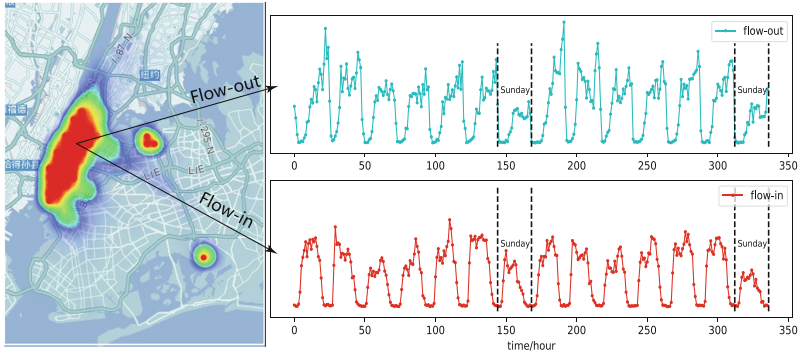
Various approaches have been developed for improving traffic flow prediction accuracy. A notable line of works [21–24] treat different regions separately and predict the flow-in and flow-out of each region using univariate time series prediction methods. However, it is well recognized that the traffic flows in nearby regions are correlated with each other, which is also known as *spatial dependencies*. To address the problem, some researches [4, 20] encapsulate spatial information as additional knowledge into the prediction model and thus achieve better performance. The main limitation of these conventional prediction methods is that they capture the spatio-temporal flow correlations using the simple linear models, which is clearly contrary to the complex scenarios in reality.

Recently, inspired by the great success of deep learning, many works adopt neural network based approaches to model complex spatio-temporal dependencies for traffic flow prediction. They naturally organize the flow-ins and flow-outs of regions during certain time period into a two-channel grid map. ST-ResNet [22] applies residual convolutional operations over the grid map to extract deep spatial features. In [21], DMVST attends to relevant temporal features extracted from historical traffic grid maps using a hierarchical network containing CNN and LSTMs [9]. All these approaches choose historical grid maps from previous time periods in a rigid manner, such as previous a few days/hours or one-week ahead maps.

To date, most of the existing methods underestimate the significance of frequency-level information contained in the past traffic flows. An important intuition is that: flow-in and flow-out sequences of a region typically involve strong frequencies. For example, consider the taxi flow-ins and flow-outs of a region in New York City over two weeks, as shown in Fig. 1. Both flow sequences generally exhibit a 24-h frequency, i.e., the flow-in and flow-out numbers within each time interval are close to those one day before. Apparently, incorporating such frequency information into the prediction model will be beneficial and vital for achieving more accurate results. However, to the best of our knowledge, none of the existing techniques are able to explicitly exploit frequency-level information for citywide traffic flow prediction.

There are two intrinsic technical challenges in developing a frequency-aware method for traffic flow prediction. First, *it is non-trivial to disclose the frequency information due to the complicated nature of traffic flows*. In particular, each traffic flow process in a region can be viewed as a composition of traffic waves

<sup>1</sup> <https://sts.didiglobal.com/views/report.html>.



**Fig. 1.** Traffic flow example. Left: a heat map of taxi traffic flow in a region of New York City at certain time. Right: the flow-in and flow-out sequences in that region over two weeks.

with different frequencies. If the traffic flow within a time interval is regarded as a random variable, it will have a large variance due to the presence of complex real-world mobility patterns. This can also be verified by sequences in Fig. 1, where no strict frequency can be derived. Second, *the frequency-level information is usually time- and space-dependent*: (i) a region at different time intervals may involve different flow frequencies, and (ii) the flow frequency in different regions can vary a lot. For example, as shown in Fig. 1, the flow frequency on Sundays is about one week rather than 24 h. Besides, the traffic flow frequencies in city centers can be quite different from those in remote areas.

To address these challenges, we have developed a novel frequency-aware spatio-temporal neural method for traffic flow prediction. Our key insight is that we can explicitly compute the frequency-level information for different regions via frequency spectrum analysis [10], and the identified frequency dynamics is useful to enhance the relevant historical flow data and suppress unrelated data for future flow prediction. Following [22], we organize the traffic flow within each time interval into a two-channel grid map, where the two channels correspond to the flow-in and flow-out respectively, and the value in each grid cell is the number of flow-ins/outs. Different from the works [21, 22] that only consider a small number of manually selected historical maps, we leverage a large number of traffic flow maps from previous time intervals as the input and perform the frequency spectrum analysis over them. We then filter irrelevant flow information according to their frequency information. Note that the filtering step is dynamically performed for different time and regions. After that, we employ 3D convolutional layers to extract the complex spatio-temporal features from the filtered input, followed by a 2D convolutional map to produce the final traffic flow in a city-wide manner.

To summarize, the major contributions of this paper are the following.

- To the best of our knowledge, this is the first attempt to explicitly explore the frequency-level information for traffic flow prediction. We propose to perform frequency spectrum analysis over a larger number of historical traffic

flow maps, and leverage the identified frequency dynamics to select the relevant historical flow information for each region and time period on future prediction.

- We develop a neural network based method named Frequency-aware Spatio-Temporal Network for traffic flow prediction. Our model is equipped with a novel frequency-based filtering mechanism to dynamically select the most relevant traffic flow data according to frequency information. We employ a 3D convolutional network to capture complex spatio-temporal correlations and the extracted spatio-temporal features are further combined with frequency features via 2D convolutions to derive the final prediction.
- We conduct extensive experiments on three real datasets. The results demonstrate that our proposed method outperforms various baseline approaches, and the frequency dynamics is useful in highlighting the relevant historical flow data for improving prediction accuracy.

The remaining of this paper is organized as follows. We review the related works in Sect. 2 and provide the preliminaries in Sect. 3. We present our frequency-aware spatio-temporal network method in Sect. 4. Experimental results are provided in Sect. 5 and we conclude this paper in Sect. 6.

## 2 Related Work

Traffic flow prediction has been studied for decades [14, 22–24]. A number of researches considered the traffic flow data in each region as a time sequence and leveraged the time series prediction methods such as Vector Autoregressive [13], Adaptive Autoregressive Integrated Moving Average [14], and logistic regression [15] for traffic flow prediction. These approaches capture complex temporal dependencies to enhance prediction accuracy, but ignore the effects of spatial correlations among traffic flows in the surrounding areas. Some works thus integrated spatio-temporal features into the prediction model. Ahn et al. [1] employed Markov random field to model the traffic flows in both spatial and temporal domains. Wu et al. [18] applied the dimension reduction technique to eliminate redundant spatio-temporal features and accelerate the overall computation process. However, all these works adopted simple linear models to capture spatio-temporal correlations, which are still insufficient to achieve high prediction accuracy.

In recent years, deep learning has achieved great success in various fields such as computer vision [7, 8, 11, 16] and natural language processing [2], which motivates many researchers to develop neural network based methods for traffic flow prediction. Zhang et al. [23] used a 2D convolutional neural network (CNN) to extract spatio-temporal features based on temporal closeness, period and seasonal trend decomposed from the original traffic flow sequences. Furthermore, Zhang et al. [22] employed the residual learning and a parametric-matrix based fusion mechanism to predict future flow more effectively. These 2D convolutional models mainly capture the non-linear spatial correlations but ignore the

temporal tendency of traffic flows. The long short-term memory (LSTM), a variant of Recurrent Neural Network (RNN), can learn long-term temporal tendency from time series data. To capture the spatio-temporal correlations, convolutional LSTM (convLSTM) [19] combines CNN with LSTM and achieves impressive performance on spatial time series forecasting. Zhou et al. [24] predicted multi-step citywide passenger demands using a novel attention based encoder-decoder framework, where attention scores are assigned to the temporal features to discriminate their importance for next step prediction. While the existing deep learning methods achieve better performance in traffic flow prediction than conventional machine learning methods, it is important to notice that traffic flow is a special type of sequential data with strong frequency dynamics, as illustrated in Fig. 1. The underlying frequency-level information involved in historical traffic flows is beneficial to highlight relevant historical flow data for each region and time period. Hence, in this paper, we propose to explicitly model the frequency dynamics from historical traffic flow data and incorporate frequency features for enhancing traffic flow prediction results.

### 3 Preliminaries

The goal of this paper is to predict citywide traffic flow, where the flow is caused by vehicle trajectories. To do this, We split the whole city area into a map of  $H \times W$  grids, which has  $H$  rows and  $W$  columns. We denote each grid by  $G_{i,j}$  where  $i \in [1, H], j \in [1, W]$ .

**Definition 1 (City Grid Map).** *We represent the target city with a rectangular area, which has  $(lon_h, lat_h)$ ,  $(lon_l, lat_l)$  for the upper left and lower right corners. Given a grid length of  $\lambda$ , we divide the whole city area into  $H \times W$  grids where*

$$H = \lceil \frac{(lat_h - lat_l)}{\lambda} \rceil, \quad W = \lceil \frac{(lon_l - lon_h)}{\lambda} \rceil$$

*The city can thus be represented as a grid map  $M = \{G_{i,j} | i \in [1, H], j \in [1, W]\}$ .*

We represent each trajectory by  $Tr = (s, e, ts, te)$ , where  $s, e$  are the start and end points, and  $ts, te$  are the start and end time. The trajectories form the traffic flow among city grids.

**Definition 2 (Citywide Traffic Flow).** *Given a city map  $M$ , a set  $\mathcal{T}$  of all the trajectories in the city, the traffic flow for a grid  $G_{i,j} \in M$  within the  $k$ -th time interval  $t_k$  includes flow-in volume  $\phi_{k,i,j}^I$  and flow-out volume  $\phi_{k,i,j}^O$ , which are defined as:*

$$\phi_{k,i,j}^I = |\{Tr \in \mathcal{T} \mid Tr.s \in G_{i,j} \wedge Tr.ts \in t_k\}| \quad (1)$$

$$\phi_{k,i,j}^O = |\{Tr \in \mathcal{T} \mid Tr.e \in G_{i,j} \wedge Tr.te \in t_k\}| \quad (2)$$

*Note that each grid involves two values for flow-in and flow-out volume, respectively. Hence, we use a 3D tensor  $\Phi \in \mathbb{R}^{H \times W \times 2}$  to represent the citywide traffic flow during the  $k$ -th time interval, which is analogy to an image with*

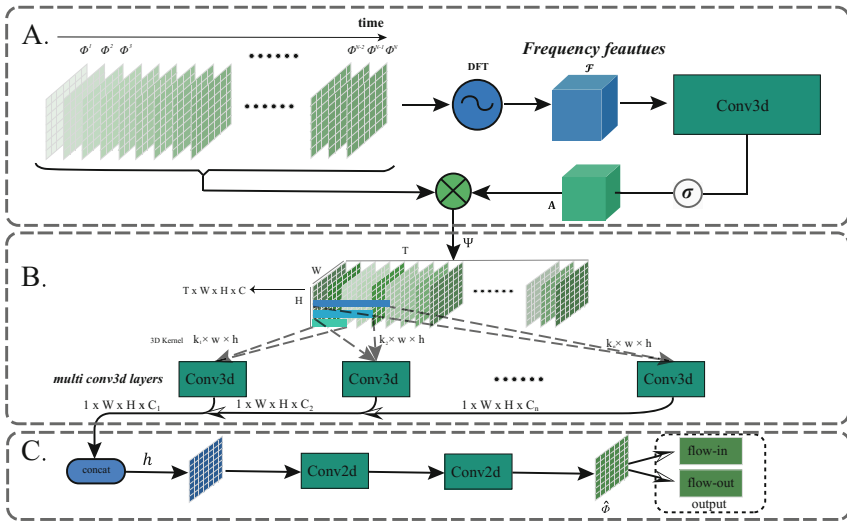


two channels. Formally, we have:

$$\Phi^k = \{[\phi_{k,i,j}^I, \phi_{k,i,j}^O] \mid 1 \leq i \leq H \wedge 1 \leq j \leq W\} \quad (3)$$

In this work, we take  $N$  citywide traffic flows for previous time periods  $k = 1, \dots, N$  as input, and try to predict the citywide traffic flow during the next time interval  $k = T + 1$ .

**Definition 3 (Traffic Flow Prediction).** *Given a  $H \times W$  city map  $M$  and a sequence of  $N$  citywide traffic flows  $\Phi = \{\Phi^k \mid k = 1 \dots, N\}$  during previous  $N$  time intervals, we aim to predict the citywide traffic flow  $\Phi^{N+1}$  for the next time period.*

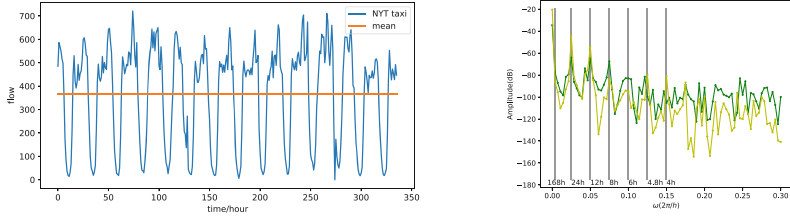


**Fig. 2.** Frequency-aware Spatio-temporal Network. (A) Frequency-based filtering module. Given a series of traffic flow maps  $\Phi^{t_1}, \dots, \Phi^{t_k}$ , we apply DFT to each grid in the map and generate filtering weight tensor. (B) Spatio-temporal convolutional module captures spatio-temporal correlation of filtered maps using multiscale 3D kernels. (C) The final output module summarizes different components and predicts the traffic flow for the next time period.

## 4 FASTNet: Frequency-Aware Spatio-Temporal Network

### 4.1 Overview

In this section, we describe our Frequency-aware Spatio-Temporal Network in detail. The transformation between time domain and frequency domain and our analysis in frequency domain will be presented at first. Then we introduce the architecture of our model. FASTNet is dedicated to using the frequency domain



(a) The flow-out of New York City taxi

(b) The spectrum of NYC taxi flow within a grid.

**Fig. 3.** The traffic flow curve and its frequency spectrum.

information and continuous long-term information to predict traffic flow for the next time period. Figure 2 depicts the architecture of our proposed Frequency-aware Spatio-temporal Network (FASTNet), which mainly contains three components.

- The first part is the frequency-based filtering module. It obtains frequency features  $\mathcal{F}$  from traffic flow tensor  $\Phi$  ( $\Phi \in \mathbb{R}^{N \times W \times H \times 2}$ ) through Discrete Fourier Transform. Frequency spectral information indicates potential periodic characteristics. We apply the 3D convolutional neural network to extract the filtering weight tensor from frequency features and then assign weights to the input tensor to obtain filtered tensor  $\Phi'$ . This filtering allows us to pay more attention to important historical flow information and ignore the noise.
- The second component can extract the spatio-temporal relationship from the filtered tensor  $\Phi'$ . Inspired by the frequency analysis, we observe that the traffic flows contain several main frequencies. And we use 3D convolutional network to model the complex spatio-temporal correlations. Different sizes of kernel in the time dimension will extract different temporal features. Finally, this module concatenate all these feature maps together and output the feature tensor  $\Psi$ .
- Finally, we summarize spatio-temporal features and predict the traffic flow  $\hat{\Phi}^{N+1}$  through a 2D convolutional network.

## 4.2 Frequency Spectrum Analysis

In the time-domain signal processing, we can transform it into the frequency domain to obtain more information through Fourier Transform. For discrete time domain signals, we use discrete-time Fourier Transform (DTFT). Supposing  $X$  is a time series, the frequency-domain spectrum is

$$\mathcal{F}_{i,j}(\omega) = \sum_{k=-\infty}^{\infty} X_k e^{-i\omega k} \quad (4)$$

where  $i = \sqrt{-1}$ . This transform needs long-term time series data and the spectrum is continuous. Specifically, for short-term time series data  $\phi_{i,j}$ , we obtain

its spectrum near time interval  $N$  by Discrete Fourier Transform (DFT). Then we will get the discrete frequency domain information, as

$$\mathcal{F}_{i,j}(\omega) = \sum_{k=1}^N \phi_{k,i,j} e^{-i\omega k} \quad (5)$$

where  $N$  is the number of the past discrete time intervals,  $\omega = \frac{2m\pi}{N}$  ( $m = 0, 1, 2, \dots, N$ ) is the frequency variable. And  $\mathcal{F}_{i,j}(\omega)$  is the spectrum near time interval  $N$  at  $G_{i,j}$ . And  $\mathcal{F}(\omega)$  can be separated into real part  $Re(\mathcal{F}(\omega))$  and imaginary part  $Im(\mathcal{F}(\omega))$ . The amplitude of each frequency  $\frac{2\pi}{\omega}$  is formulated as:

$$|\mathcal{F}_{i,j}(\omega)| = \sqrt{Re(\mathcal{F}_{i,j})^2 + Im(\mathcal{F}_{i,j})^2} \quad (6)$$

To analysis this frequency spectrum, we transform the amplitude to decibel as below

$$|\mathcal{F}_{i,j}(\omega)|_{dB} = 20 * \log(|\mathcal{F}_{i,j}(\omega)|) \quad (7)$$

Figure 3a shows the curve of the two-week flow-outs in New York City. The traffic flow is very similar to signals with multiple frequencies and contains some noise. We can map it into the frequency domain to analyze its components and find its patterns. Obviously, the taxi traffic data shows a certain periodicity, such as 24 h and a week, although not strictly consistent. From a signal point of view, the traffic flow can be seen as a superposition of many sub-flows with different frequencies. To predict the next traffic flow volume, we may use the frequency domain information in the time dimension to make predictions.

Applying DFT to traffic flow volume data, we can obtain the spectrum at each grid. We remove half spectrum of NYC taxi flow due to the symmetry of DFT. As Fig. 3b shows, there are many reasonable periods that have higher amplitudes than others, such as 24 h, 12 h, 8 h and etc. However, the amplitudes of the main components within different grids can vary greatly, indicating that different regions have different frequency characteristics. We need to capture these differences in our model. Specifically, by leveraging these features, we can infer which parts of historical data are more important to predict the next traffic flow and enlighten more appropriate spatio-temporal network kernel sizes. In our experiments, we aggregate all the traffic flow of each grid and apply DFT to these series to obtain their frequency spectrum. In frequencies where the amplitude exceeds the noise, we choose several peak frequencies as the *main frequency components*.

### 4.3 Frequency-Based Filtering

Once we extract the frequency information from input historical data, we need to determine which parts of data in the fragment are significantly important for the prediction. We purpose a frequency-based filtering mechanism for this purpose. Our frequency-based filtering mechanism generates vectors  $\mathbf{a}_{i,j} = \{\mathbf{a}_{i,j}^I, \mathbf{a}_{i,j}^O\}$  for each grid and gives more importance to the most relevant historical data.

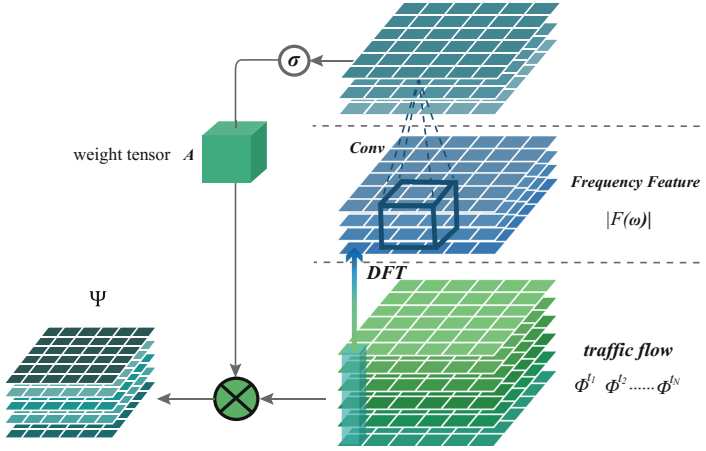


Fig. 4. Frequency-based filtering mechanism

**Filtering Mechanism:** As shown in Fig. 4, our filtering mechanism consists of three steps, which dynamically determines the most relevant flow to the prediction of each grid. All the filtering weights are obtained from traffic flow using DFT and convolutional layers. First, suppose we use the traffic flow data of the nearest  $N$  time intervals to predict the traffic flow in the next time period. For each grid  $G_{i,j}$  in the city map, we apply Eq. (5) to get the spectrum  $|\mathcal{F}_{i,j}(\omega)|, \omega = \frac{2m\pi}{N}, m = 0, 1, \dots, N/2$ , which is a 4D tensor ( $2 \times W \times H \times N/2$ ) including flow-in and flow-out frequency features. Second, we develop multiple 3D convolutional layers with kernel size  $1 \times s_1 \times s_2$  ( $s_1$  and  $s_2$  are the kernel parameters), which summarize neighbors' spectrum and extract the filtering weight tensor from the spectrum. In this step, our model generates the input importance of each time interval. Finally, after multiple convolution layers and sigmoid function, we obtain the filtering weight tensor  $A \in \mathbb{R}^{N \times W \times H \times 2}$ . For flow-in and flow-out at  $G_{i,j}$ , we have filtering vectors  $\mathbf{a}_{i,j}^I$  and  $\mathbf{a}_{i,j}^O$ , respectively. We multiply the filtering tensor  $A$  with input tensor by hadamard product (denoted by  $\circ$ ). The key equations of our filtering mechanism are formulated as follows:

$$\mathcal{F}_{i,j}(\omega) = \sum_{k=1}^N \phi_{k,i,j} e^{-i\omega k} \quad \omega = \frac{2\pi m}{N}, m = 0, 1, \dots, \frac{N}{2} \quad (8)$$

$$\mathbf{A} = \sigma(f(W_{f2} * f(W_{f1} * |\mathcal{F}(\omega)| + b_{f1}) + b_{f2})) \quad (9)$$

$$\Psi = \mathbf{A} \circ \Phi^{t_k} \quad t = 1, 2, \dots, N \quad (10)$$

where  $\mathcal{F} \in \mathbb{R}^{2 \times W \times H \times (N/2+1)}$ ,  $|\cdot|$  is the module of each number in a matrix,  $*$  denotes a convolution operation and  $f(\cdot)$  represents a non-linear activation function.

#### 4.4 Spatio-Temporal Convolutional Module

In addition to frequency filtering, traffic flow typically has strong spatio-temporal dependency. Previous works [22, 23] used residual convolutional networks or LSTM [21] to capture spatio-temporal correlations, which underutilized the full historical data or failed to capture longtime dependency. To overcome these deficiencies, we extend the spatial CNN to spatio-temporal convolutional neural network (**ST-CNN**), and employ 3D convolutional layers to model the spatio-temporal correlations.

From the spectrum analysis we can know that there are many important frequency components in the traffic flow. For each frequency component, we can extract the features by setting the kernel size and convolution stride according to its corresponding scale. For different frequency components, we set up multiple convolution kernels of different sizes to extract spatio-temporal features of different scales. Our spatio-temporal convolutional component can leverage data from hundreds of time intervals. Figure 2b shows the structure of this module. We use multiple 3D convolution kernels, which have the same size with stride in temporal dimension. The size of each kernel is  $k \times w \times h$ , which means that we perform a convolution operation every  $k$  data points of inputs without overlapping. According to the frequency spectrum, there are several *main frequency components*, which are used as the 3D kernel sizes in temporal dimension. That is to say, our model uses some significant frequencies to set the kernel size in time space. The input map  $\Psi$  is 4D tensor,  $\Psi \in \mathbb{R}^{T \times W \times H \times 2}$  and the output is  $h^l$  for layer  $l$ . Due to the excellent performance of our frequency domain filtering mechanism and the 3D convolutional network, we only need to use a few convolutional layers to extract complex spatio-temporal features. Finally, we concatenate all the features together. The first layer and output can be formulated as follows:

$$h_k^1 = f(W_k * \Psi + b_k) \quad (11)$$

$$h = h_1^L \oplus h_2^L \oplus \dots \oplus h_n^L \quad (12)$$

where  $h_k^1$  is the output of first convolutional layer with  $k$ -th kernel.  $\oplus$  donates concatenate operation.  $n$  is the number of kernels. And  $h$  is the final output map of this component.

#### 4.5 Prediction Component

The final prediction component utilize the features  $h$  extracted by spatio-temporal convolutional module to forecast the traffic flow  $\hat{\Phi}^{N+1}$  in next interval. We employ two convolutional layers to compute the prediction results  $\hat{\Phi}^{N+1}$ , which are defined as follows:

$$\hat{\Phi}^{N+1} = f(W_2 * (f(W_1 * h) + b_1) + b_2) \quad (13)$$

where  $W_1$ ,  $W_2$ ,  $b_1$  and  $b_2$  are parameters to be learned.

**Table 1.** Datasets

Dataset	Time span	Time interval	Grid map size
NYCTaxi	01/01/2014–06/30/2015	1 h	48 × 32
NYCBike	04/01/2014–09/30/2014	1 h	16 × 8
TaxiCD	11/01/2016–11/30/2016	15 min	48 × 48

## 4.6 Optimization

Our FASTNet model predicts the traffic flow in the next time interval with  $N$  previous traffic flows. Our model can be trained by the following loss function.

$$\mathcal{L}(\theta) = \|\Phi^{N+1} - \hat{\Phi}^{N+1}\|_2^2 \quad (14)$$

where  $\theta$  denotes all the trainable parameters.

## 5 Experiments

In this section, we conduct extensive experiments on three traffic datasets to evaluate the performance of our model. We also analyze the frequency-based filtering mechanism and show the effect of the number of input time intervals.

### 5.1 Datasets and Settings

**Datasets.** We use three traffic datasets: TaxiNYC<sup>2</sup>, BikeNYC<sup>3</sup> and TaxiCD<sup>4</sup>. The details of all the datasets are presented in Table 1. Regarding people’s actual travel distances, we set the grid size of each city as 1 km × 1 km. We set the time interval to be 1 h for TaxiNYC and BikeNYC, and 15 min for TaxiCD. We remove the records in abnormal positions and compute traffic flow volume during each time interval in each map. Then we eliminate the outer grids that have few records. After that, we get the predictable map segmentations of each dataset. We choose the first 80% of the samples as training data, the following 10% for validation, which is used for parameter tuning, and the remaining for test.

**Experimental Settings.** Our experiments are conducted on a server with two NVIDIA Titan Xp GPUs. We implement our model and other baselines in Python and the third-party library mainly including tensorflow [6], xgboost and statsmodels. Specifically, time interval lengths of input( $N$ ) respectively are 168, 168 and 672 for TaxiNYC, BikeNYC and TaxiCD, respectively. We truncate the half spectrum information from DFT as the input for filtering module. We use two convolutional layers to generate filtering weight tensor. The kernel sizes are

<sup>2</sup> [http://www.nyc.gov/html/about/trip\\_record\\_data.shtml](http://www.nyc.gov/html/about/trip_record_data.shtml).

<sup>3</sup> <https://citibikenyc.com/system-data>.

<sup>4</sup> <https://gaia.didichuxing.com>.

**Table 2.** RMSE comparison with different baselines

Dataset	HA	VAR	ARIMA	RG	XGBoost	ST-ResNet	Conv-LSTM	FASTNet-N	FASTNet
NYCTaxi	22.85	13.15	18.44	16.21	15.86	9.37	8.55	7.82	<b>7.34</b>
NYCBike	8.53	7.19	7.21	8.76	6.11	5.12	5.05	5.01	<b>4.60</b>
CDTaxi	1.62	1.66	1.76	1.85	1.58	1.57	1.50	1.39	<b>1.31</b>

both  $(1 \times 3 \times 3)$ . We choose sigmoid function to transform the weights to  $(0, 1)$ . For 3D spatio-temporal CNN, we use two 3D convolutional layers to extract the complex spatio-temporal correlation. The sizes of kernel in first convolutional layer are  $(8, 3, 3)$ ,  $(12, 3, 3)$ ,  $(24, 3, 3)$  except for TaxiCD. For TaxiCD, the kernel sizes are  $(16, 3, 3)$ ,  $(32, 3, 3)$ ,  $(48, 3, 3)$ ,  $(96, 3, 3)$ . The number of each type kernel is 64. The final prediction component consists of two convolutional layers. The kernel sizes of predicting layers are both  $(3 \times 3)$ . We use RMSPropOptimizer [17] as optimizer and set the learning rate to be 0.0002.

## 5.2 Measurement and Baseline Methods

We measure the performance of different methods by Root MeanSquare Error (RMSE), which is defined as follows:

$$RMSE = \sqrt{\frac{1}{Q} \sum_i \sum_j \sum_k (\phi_{k,i,j} - \hat{\phi}_{k,i,j})^2} \quad (15)$$

where  $Q = \xi \times W \times H \times 2$ , and  $\xi$  is the number of samples.

We compare FASTNet with seven baseline methods, which are Historical Average (HA), Vector Auto Regression (VAR) [5], Auto-Regressive Integrated Moving Average (ARIMA), Auto-Regressive Integrated Moving Average (ARIMA), Ridge Regression, XGBoost [3], ST-ResNet [22], ConvLSTM [19].

## 5.3 Performance Comparison

Table 2 shows the performance of our model compared to all the baselines on three datasets. Result differences between three datasets are caused by different total traffic flow volumes. The rooted mean square values of the traffic flow volume in three datasets are 91.46 (TaxiNYC), 20.89 (BikeNYC) and 6.06 (TaxiCD), respectively. **FASTNet** and **FASTNet-N** (without Filtering) achieve best performance with the lowest RMSE than the other baseline methods. On three datasets, our model improves RMSE by 14.2%, 8.9% and 12.7% than ST-ResNet. We can also observe that FASTNet performs better than FASTNet-N, which does not have the filtering mechanism. We also notice that the performance of HA method is not too bad on these datasets, which reveals that the traffic flow data presents periodicity. The VAR and ARIMA do not achieve impressive performance, because these models only use a small

**Table 3.** Comparison with variants

Dataset	FASTNet-K20-K16-K6	FASTNet-K24	FASTNet-K24-K12	FASTNet
NYCTaxi	7.71	7.73	7.41	<b>7.34</b>
NYCBike	4.87	4.75	4.64	<b>4.60</b>

amount of historical data and fail to model complex spatio-temporal correlations. The regression methods (Ridge Regression and XGBoost) do not perform well, because these linear models cannot model the non-linear spatio-temporal correlations very well. ConvLSTM and ST-ResNet perform better than classical machine learning methods. But they all ignore the long-term (more than one hundred time intervals) continuous dependency of the flow data. For ST-ResNet, the historical data is split into pieces. And only part of these data is used as input, which may lose some important information and break the periodical information. FASTNet not only assigns weights to the important historical data, but also model spatio-temporal correlations with long sequences.

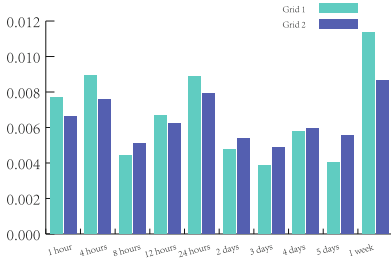
#### 5.4 Influence of Different Kernel Sizes in ST-CNN

We analyze the influence of different kernel sizes in temporal dimension. It is intuitive that frequency components of traffic flow come from the cycle of daily life. The kernel size in temporal dimension is very important for prediction. The first layer in our 3D ST-CNN extract long time features over multiple cycles using different 3D kernels. We design several variants of FASTNet for NYCTaxi and NYCBike datasets by setting the first layer of 3D ST-CNN with different kernels according to the *main frequency components*:

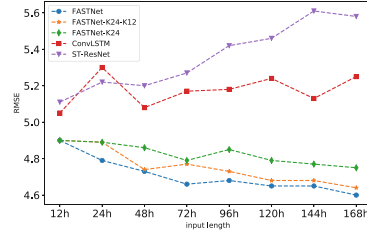
- **FASTNet-K24**: Kernel is set to  $24 \times 3 \times 3$ .
- **FASTNet-K24-K12**: Kernels are set to  $24 \times 3 \times 3$  and  $12 \times 3 \times 3$ .
- **FASTNet-K20-K16-K6**: Kernels are set to  $20 \times 3 \times 3$ ,  $16 \times 3 \times 3$  and  $6 \times 3 \times 3$ .
- **FASTNet**: Kernels are set to  $24 \times 3 \times 3$  and  $12 \times 3 \times 3$ ,  $8 \times 3 \times 3$ .

As shown in Table 3, thanks to the frequency-based filtering mechanism and spatio-temporal architecture, all the variants achieve lower RMSE than the baseline methods. It is obvious that FASTNet performs consistently better than other variants on NYCTaxi and NYCBike. FASTNet-K20-K16-K6’s kernel sizes in time dimension are not fit for the *main frequency components* and therefore performs poorly. Compared to FASTNet-24K, FASTNet-24K-12K achieves 4.1% and 2.3% improvement. Furthermore, FASTNet with three relevant kernels achieves 1% lower RMSE than FASTNet-K24-K12 on both NYCBike and NYCTaxi. It shows that the 12-h periods and 8h periods are both important for prediction. In general, the more related kernels we add, the more prediction accuracy we can gain.





**Fig. 5.** Filtering weights of two grids on NYCBike.



**Fig. 6.** RMSE of different input lengths on NYCBike.

## 5.5 Filtering Weights Analysis

Our frequency-based filtering mechanism can enhance the most relevant part of input and filter irrelevant frequency information. Here, we will explore the functionality of the filtering mechanism by visualizing part of the filtering weights. At different grids, the filtering mechanism can assign different importance to historical data according to the frequency features. We select two grids in NYCBike dataset. Grid 1 is closer to the city center and the daily traffic flow volume is very large during a week. Grid 2 is remote from center and the traffic flow during a week is very uneven. Figure 5 shows the comparison between these two grids on NYCBike map. We extract the weights of 10 time intervals. Higher coefficient means the data at this time interval is more important and we want to enhance it. It can be seen that the Grid 1 gives more attention to the near data that in a day such as 1 h, 4 h, 24 h. Also, Grid 1 takes the data 1 week ago as the most important part. Obviously, the weights of Grid 2 is different from that of Grid 1, which is more balanced. Compared with Grid 1, Grid 2 is interested in the data for one-day basis. This is because Grid 1 is much far away from downtown than Grid 2, and Grid 2 has more balanced traffic flow.

## 5.6 Influence of Sequence Length

We now study the effect of the input length. Our FASTNet can model spatio-temporal correlations of very long sequence. It excels at discovering patterns that have periodic variations over time. Previous deep-learning methods can only handle short input lengths. With the increase of input length, the performance of LSTM may decrease because of gradient vanishing. To analyze the influence of sequence length, we set the input length to be one of  $\{12, 24, 48, 72, 96, 120, 144, 168\}$ . Figure 6 shows the influence of different input lengths on NYCBike dataset. Our FASTNet model performs better as the sequence length increases. ST-ResNet uses the incomplete data as input, which fail to extract the traffic flow features in the distant past. ConvLSTM has much stable performance than ST-ResNet, even though it does not have much performance gain with more inputs. In FASTNet, the frequency-based filtering mechanism can filter the data

over a long time and the 3D spatio-temporal convolutional module treats the input data as small fragments, which share the parameters to extract features from longer input sequences. Overall, our model can incorporate with hundreds of time-interval inputs and achieve better performance.

## 6 Conclusion

In our paper, we analyze the frequency domain characteristics of traffic flow data for traffic flow prediction. Based on the explored frequency characteristics, we propose a frequency-aware spatio-temporal network (FASTNet), which leverages frequency spectrum information to dynamically filter historical data and skillfully integrate 3D convolution to model spatio-temporal correlations. Our model is able to take the advantages of historical data with longer lengths towards better prediction performance. We also evaluate our method on three traffic flow datasets, i.e., TaxiNYC, BikeNYC, CDTaxi. Extensive experiments demonstrate the proposed method outperforms all the seven baseline methods in prediction accuracy. The experiments on various input lengths verify that our model is able to effectively extract long-term dependency features.

**Acknowledgements.** This work is supported by the National Key Research and Development Program of China (No. 2018YFC0831604). Yanyan Shen is in part supported by NSFC (No. 61602297). Yanmin Zhu is in part supported by NSFC (No. 61772341, 61472254) and STSCM (No. 18511103002). Yuting Chen is in part supported by NSFC (No. 61572312) and Shanghai Municipal Commission of Economy and Informatization (No. 20170101052).

## References

1. Ahn, J., Ko, E., Kim, E.Y.: Predicting spatiotemporal traffic flow based on support vector regression and Bayesian classifier. In: Fifth IEEE International Conference on Big Data and Cloud Computing, BDCLOUD 2015, Dalian, China, 26–28 August 2015, pp. 125–130 (2015). <https://doi.org/10.1109/BDCLOUD.2015.64>
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint [arXiv:1409.0473](https://arxiv.org/abs/1409.0473) (2014)
3. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint [arXiv:1412.3555](https://arxiv.org/abs/1412.3555) (2014)
4. Deng, D., Shahabi, C., Demiryurek, U., Zhu, L., Yu, R., Liu, Y.: Latent space model for road networks to predict time-varying traffic. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1525–1534. ACM (2016)
5. El-Yaniv, R., Faynburd, A.: Autoregressive short-term prediction of turning points using support vector regression. CoRR abs/1209.0127 <http://arxiv.org/abs/1209.0127> (2012)
6. Girija, S.S.: Tensorflow: large-scale machine learning on heterogeneous distributed systems (2016)
7. Girshick, R.: Fast R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1440–1448 (2015)

8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: *Computer Vision and Pattern Recognition*, pp. 770–778 (2016)
9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.* **9**(8), 1735–1780 (1997)
10. Kay, S.M., Marple, S.L.: Spectrum analysis—a modern perspective. *Proc. IEEE* **69**(11), 1380–1419 (1981)
11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks, pp. 1097–1105 (2012)
12. Li, S., Li, W., Cook, C., Zhu, C., Gao, Y.: Independently recurrent neural network (IndRNN): building a longer and deeper RNN. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5457–5466 (2018)
13. Lu, Z., Zhou, C., Wu, J., Jiang, H., Cui, S.: Integrating granger causality and vector auto-regression for traffic prediction of large-scale wlns. *TIIS* **10**(1), 136–151 (2016). <https://doi.org/10.3837/tiis.2016.01.008>
14. Shekhar, S., Williams, B.: Adaptive seasonal time series models for forecasting short-term traffic flow. *Transp. Res. Rec. J. Transp. Res. Board* **2024**, 116–125 (2008)
15. Sun, H., Liu, H., Xiao, H., He, R., Ran, B.: Use of local linear regression model for short-term traffic forecasting. *Transp. Res. Rec. J. Transp. Res. Board* **1836**, 143–150 (2003)
16. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, inception-resnet and the impact of residual connections on learning. In: *AAAI*, vol. 4, p. 12 (2017)
17. Tieleman, T., Hinton, G.: RMSprop gradient optimization (2014). [http://www.cs.toronto.edu/tijmen/csc321/slides/lecture\\_slides\\_lec6.pdf](http://www.cs.toronto.edu/tijmen/csc321/slides/lecture_slides_lec6.pdf)
18. Wu, Y., Chen, F., Lu, C., Yang, S.: Urban traffic flow prediction using a spatio-temporal random effects model. *J. Intellig. Transp. Syst.* **20**(3), 282–293 (2016). <https://doi.org/10.1080/15472450.2015.1072050>
19. Xingjian, S., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.K., Woo, W.C.: Convolutional LSTM network: a machine learning approach for precipitation nowcasting. In: *Advances in Neural Information Processing Systems*, pp. 802–810 (2015)
20. Xu, J., Deng, D., Demiryurek, U., Shahabi, C., Van Der Schaar, M.: Context-aware online spatiotemporal traffic prediction. In: *2014 IEEE International Conference on Data Mining Workshop (ICDMW)*, pp. 43–46. IEEE (2014)
21. Yao, H., et al.: Deep multi-view spatial-temporal network for taxi demand prediction. arXiv preprint [arXiv:1802.08714](https://arxiv.org/abs/1802.08714) (2018)
22. Zhang, J., Zheng, Y., Qi, D.: Deep spatio-temporal residual networks for citywide crowd flows prediction. In: *AAAI*, pp. 1655–1661 (2017)
23. Zhang, J., Zheng, Y., Qi, D., Li, R., Yi, X.: DNN-based prediction model for spatio-temporal data. In: *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, p. 92. ACM (2016)
24. Zhou, X., Shen, Y., Zhu, Y., Huang, L.: Predicting multi-step citywide passenger demands using attention-based neural networks, pp. 736–744 (2018)



# Efficient Algorithms for Solving Aggregate Keyword Routing Problems

Qize Jiang<sup>1,2,3</sup>, Weiwei Sun<sup>1,2,3(✉)</sup>, Baihua Zheng<sup>4</sup>, and Kunjie Chen<sup>1,2,3</sup>

<sup>1</sup> School of Computer Science, Fudan University, Shanghai, China  
{qzjiang18, wwsun, chenkunjie}@fudan.edu.cn

<sup>2</sup> Shanghai Key Laboratory of Data Science, Fudan University, Shanghai, China

<sup>3</sup> Shanghai Institute of Intelligent Electronics and Systems, Shanghai, China

<sup>4</sup> School of Information Systems, Singapore Management University, Singapore,  
Singapore  
bhzheng@smu.edu.sg

**Abstract.** With the emergence of smart phones and the popularity of GPS, the number of *point of interest (POIs)* is growing rapidly and spatial keyword search based on POIs has attracted significant attention. In this paper, we study a more sophisticated type of spatial keyword searches that considers multiple query points and multiple query keywords, namely *Aggregate Keyword Routing (AKR)*. AKR looks for an aggregate point  $m$  together with routes from each query point to  $m$ . The aggregate point has to satisfy the aggregate keywords, the routes from query points to the aggregate point have to pass POIs in order to complete the tasks specified by the task keywords, and the result route is expected to be the optimal one among all the potential results. In order to process AKR queries efficiently, we propose effective search algorithms, which support different aggregate functions. A comprehensive evaluation has been conducted to evaluate the performance of these algorithms with real datasets.

**Keywords:** Aggregate keyword query · Query processing ·  
Route planning

## 1 Introduction

The emergence of smart phones and the popularity of GPS have spawned a revolution in mobile location-based capabilities. Many users of mobile Apps are voluntary information contributors. For example, many mobile Apps that provide *location-based services* allow users to upload and update the description of locations, e.g., Foursquare<sup>1</sup>. With the help of these *User Generated Contents*, the number of *point of interest (POIs)* is growing rapidly. Accordingly, the searches conducted by users are no longer only about spatial features but also textual contents. A *spatial keyword query* that aims at finding a POI which is closest

<sup>1</sup> <https://www.foursquare.com>.

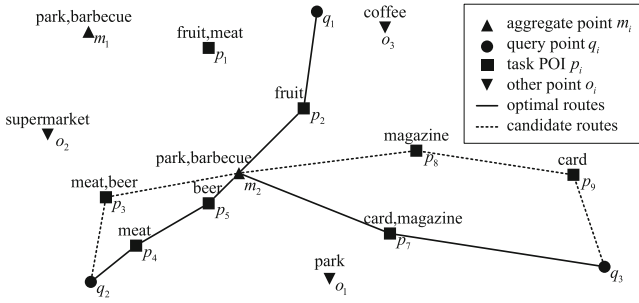


Fig. 1. Example of aggregate keyword routing

to the query point and meanwhile is relevant to the requested keywords is one example. Besides, *Aggregate Nearest Neighbor (ANN)* query [12], which finds the aggregate point with the smallest aggregate distance for a given set of spatial points and a given set of query points, is also a hot research topic.

In this paper, we study a more sophisticated type of spatial keyword searches that considers multiple query points and multiple query keywords. Before we formally introduce the query, let’s consider the following scenario, as detailed in Example 1. Function  $\text{keyword}(P)$  takes in a set of POIs  $P$  as input and returns the set of keywords associated with any POI  $p \in P$ , and function  $\text{poi}(R)$  takes in a set of routes  $R$  as input and returns the set of POIs passed by any single route  $r \in R$ .

**Example 1.** *Alex, Bob, and Carol want to organize a barbecue in a park. On their ways to the barbecue venue, they need to purchase beer, meat, and fruit for the barbecue and magazines and cards to pastime during the barbecue. Assume Fig. 1 plots all the POIs. In order to facilitate planing of the barbecue, they want to conduct a search that takes in their home locations (i.e.,  $q_1, q_2$  and  $q_3$  in Fig. 1), and two set of keywords, namely task keywords  $\kappa_t = \{\text{beer, meat, fruit, magazine, card}\}$  and aggregate keywords  $\kappa_a = \{\text{park, barbecue}\}$ , as input, and output a gathering point  $m$  and three routes  $r_1, r_2, r_3$  from their home locations to  $m$  respectively. To be more specific, the query is expected to satisfy following three conditions: (i) the gathering point  $m$  needs to satisfy the textual requirement represented by  $\kappa_a$  such as  $m_1$  and  $m_2$  in Fig. 1, i.e.,  $\text{keyword}(\{m\}) \supseteq \kappa_a$ ; (ii) the set of POIs passed by three routes, denoted by  $\text{poi}(\cup_{i=1}^3 r_i)$ , is able to satisfy the textual requirement represented by  $\kappa_t$ , i.e.,  $\text{keyword}(\text{poi}(\cup_{i=1}^3 r_i)) \supseteq \kappa_t$ ; and (iii) for any other answer set  $\langle m', \cup_{i=1}^3 r'_i \rangle$  that satisfies the above two conditions, the aggregate distance (e.g., avg, max, sum) of  $\cup_{i=1}^3 r_i$  does not exceed that of  $\cup_{i=1}^3 r'_i$  in order to guarantee that the search returns the optimal solution. For example, if **maximum** is the selected aggregate distance function,  $m_2$  and the three solid-line routes form the solution.*

The search conducted in Example 1 considers spatial condition represented by the set of query points  $Q$  and textual conditions represented by the two sets of keywords, denoted as  $\kappa_a$  and  $\kappa_t$ , and expects a single answer point, together with routes from each individual query point to the answer point. The textual

condition requires the answer point to satisfy one set of keywords  $\kappa_a$ , and the POIs passed by the routes to cover the other set of keywords  $\kappa_t$ ; while the spatial condition requires the routes to provide the optimal solution for certain given distance functions. Accordingly, this spatial keywords search is named as *Aggregate Keyword Routing (AKR)*.

AKR is challenging and time consuming. It expects an aggregate point  $m$  and a set of routes  $\cup_i r_i$  from each query point to  $m$  as the result, but there are a great number of such possible routes. Consequently, the search space for the qualified routes for a given aggregate point is very large. Not to mention that the search space for the aggregate points could be large too. In fact, AKR query is NP-Hard.

The exact and approximate algorithms to solve the AKR problem are available, i.e., Task Assignment and Routing (TAR) and Center Based Assignment (CBA) proposed in [1]. Our solution outperforms TAR in terms of efficiency, while it achieves a higher accuracy than CBA. Furthermore, we take into account the **average** aggregate distance function, which was not discussed in [1], and extend all algorithms to support the new function.

In summary, this paper makes following major contributions.

- We propose novel search algorithms to process AKR queries, namely Tree Expansion (TE) and TE-ext, which are efficient and effective.
- Our algorithms support both **maximum** and **average** aggregate distance functions, and we extend the existing algorithms proposed in [1] to support the **average** aggregate distance function too.
- We conduct comprehensive experimental study to evaluate the performance of our algorithms with real POI data in different scales. The results demonstrate the efficiency and correctness of our algorithms in different aggregate functions and data scales.

The remaining of this paper is organized as follows. Section 2 reviews the related work. Section 3 formally defines the problem of *Aggregate Keyword Routing*. In Sect. 4, we present algorithm TE and its extension. For structural clarity, we only consider **maximum** as the aggregate distance function in Sect. 4; while we present the variances of algorithms to support AKR query using **average** as the aggregate distance function in Sect. 5. In Sect. 6, we analyze the worst-case time complexity of the algorithms. Section 7 reports our experimental evaluation results. Finally, we conclude our paper in Sect. 8.

## 2 Related Work

AKR query combines aggregate nearest neighbor (ANN) with semantic similarity. In the following, we mainly review existing works related to ANN query, spatial keyword query and AKR query.

### 2.1 ANN Query and Spatial Keyword Query

Aggregate Nearest Neighbor (ANN) is a traditional problem. In the literature, ANN and many of its variances have been studied. [12] proposes the ANN query,

and analyzes the average function situation. [13] considers maximum, minimum, and average aggregate distance functions under R-tree. [5] explores the ANN query in high dimension. [10,11] solve the ANN query without indexing. [14] incorporates Voronoi diagrams into R-tree to take advantages of the strength from both structures. [15,19] study ANN query in road networks. [4] finds a group from aggregate points and minimizes the total distance from query points to group. [16,17] explore the merged aggregate nearest neighbor query. [23] considers road network's Voronoi graph and solves ANN problems for both sum and maximum functions. [9] studies ANN in uncertain databases, and proposes effective pruning methods to reduce the search space.

Spatial keyword queries have also been well studied recently. [3] proposes IR<sup>2</sup>-tree, which integrates R-tree and signature files; and [21] proposes bR\*-tree that combines R\*-tree with bitmap and keyword MBR. IR-tree [2,8] attaches each MBR with inverted lists, and supports ranking queries in respect to both spatial proximity and semantic similarity. [7] studies top- $k$  aggregate nearest keyword query. [22] studies aggregate keyword nearest neighbor query, which finds the nearest neighbor with certain keywords. [6] considers direction-aware spatial keyword search, which considers keyword's direction. [20] proposes IL-Quadtree, which is based on inverted index and the linear quadtree, and develops an efficient algorithm to tackle top- $k$  spatial keywords search. [18] considers the multi-approximate keyword routing problem.

## 2.2 AKR Query

The AKR problem was first studied in [1]. The authors proposed an exact algorithm and an approximate algorithm, namely Task Assignment and Routing (TAR) and Center Based Assignment (CBA) respectively.

TAR is a two-phase algorithm, which consists of *Search and Assignment Phase* and *Task Finishing Phase*. In Search and Assignment Phase, existing ANN search algorithms are used to determine access order, i.e., the priority of the potential aggregate points, where a technique called *early termination* is applied to reduce the number of possible points. Then candidate POIs on the routes from each query point to the aggregate point are determined. In Task Finishing Phase, the algorithm searches for optimal routes and prunes the useless routes, taking advantages of the heap data structure.

CBA, on the other hand, is an approximation algorithm. It first locates an aggregate point  $m$  that is close to all query points. Afterwards, a nearest neighbor search around  $m$  generates a set of POIs which need to be passed. Finally, the result routes are computed through a heuristic approach, which starts from the shortest path and then inserts a POI into the path.

## 3 Problem Formulation

In the context of this paper, a POI  $p$  is associated with its location and a set of keywords  $\kappa(p) = \{\delta_1, \delta_2, \dots, \delta_k\}$  with  $k = |\kappa(p)|$ . An *Aggregate Keyword Routing*

**Table 1.** Frequently used notations

Notation	Explanation
$P$	set of POIs
$Q$	set of query points
$r_{q,m}$	a route $\langle q, p_1, \dots, p_x, m \rangle$ from point $q$ to point $m$
$R_{Q,m}$	a route set $\{\cup_{\forall q \in Q} r_{q,m}\}$ from query points in $q$ to $m$
$R_{Q,m}^0$	the shortest route set from $Q$ to $m$ without passing any other point
$L(r)$	length of route $r$
$L_f(R)$	length of route set $R$ with aggregate distance function $f$
$L_{max}(R)$	the maximum aggregate distance of $R$ , i.e., $\text{MAX}_{\forall r \in R} L(r)$
$L_{avg}(R)$	the average aggregate distance of $R$ , i.e., $\frac{1}{ R } \sum_{\forall r \in R} L(r)$
$\alpha$	the aggregate point set
$\kappa_a$	the aggregate keyword set
$\kappa_t$	the task keyword set
$\delta$	a single keyword
$\kappa(R)$	set of task keywords covered by set $R$
$m$	a candidate aggregate point

*Query* (AKR) takes in three parameters as input, i.e., a query point set  $Q$ , a task keyword set  $\kappa_t$ , and an aggregate keyword set  $\kappa_a$ . For a given task keyword set  $\kappa_t$ , a POI  $p$  is considered as a *task* POI iff  $\kappa(p) \cap \kappa_t \neq \emptyset$ . Table 1 lists the notations that will be frequently used in the rest of this paper.

In the following, we first present the terms of *route*, *route set*, and the *aggregate distance of a route set* in Definitions 1, 2, and 3 respectively. We then introduce *candidate result of an aggregate keywords routing query* in Definition 4 and present the formal definition of AKR in Definition 5.

**Definition 1.** A route  $r_{s,m} = \langle s, p_1, \dots, p_x, m \rangle$  is a point sequence that starts from point  $s$ , goes sequentially through  $p_1$  to  $p_x$  and ends at point  $m$ . We denote length of the route as  $L(r_{s,m})$  and the keyword set covered by the route as  $\kappa(r_{s,m})$ , i.e.,  $\kappa(r_{s,m}) = \cup_{\forall p_i \in r_{s,m}} \kappa(p_i)$ .

**Definition 2.** Given a query point set  $Q = \{q_1, q_2, \dots, q_n\}$ , we use notation  $R_{Q,m}$  to represent a route set  $\{r_{q_1,m}, r_{q_2,m}, \dots, r_{q_n,m}\}$  and notation  $\kappa(R_{Q,m})$  to capture the keyword set covered by any route in  $R_{Q,m}$ , i.e.,  $\kappa(R_{Q,m}) = \cup_{\forall r \in R_{Q,m}} \kappa(r)$ .

**Definition 3.** The length of  $R_{Q,m}$  is marked as  $L_f(R_{Q,m})$ , dependent on the given aggregate distance function  $f$ . For example, if **maximum** is the aggregate distance function, we have  $L_{max}(R_{Q,m}) = \text{MAX}_{r_{q,m} \in R_{Q,m}} L(r_{q,m})$ ; if **average** is the aggregate distance function, we have  $L_{avg}(R_{Q,m}) = \frac{1}{|Q|} \sum_{r_{q,m} \in R_{Q,m}} L(r_{q,m})$ .

**Definition 4.** Given an Aggregate Keyword Routing (AKR) query  $\langle Q, \kappa_a, \kappa_t \rangle$ , an aggregate point  $m$  and a route set  $R_{Q,m}$  form a candidate AKR result



$\langle m, R_{Q,m} \rangle$  if and only if  $\kappa_a \subseteq \kappa(m)$  and  $\kappa_t \subseteq \kappa(R_{Q,m})$ . We denote the complete set of candidate AKR results as  $A_R$ . As the aggregate point  $m$  could be derived from  $R_{Q,m}$ , we use  $R_{Q,m}$  to represent a candidate AKR result but skip  $m$  for brevity.

**Definition 5.** Given an aggregate distance function  $f$ , an Aggregate Keyword Routing (AKR) query  $\langle Q, \kappa_a, \kappa_t \rangle$  is to locate the candidate result set  $R_{result}$  with the minimum  $L_f(R_{result})$  value, i.e.,  $R_{result} = \arg \min_{R_{Q,m} \in A_R} L_f(R_{Q,m})$ .

As mentioned above, processing of AKR query is very time consuming. In fact, it is NP-Hard, as presented in Theorem 1.

**Theorem 1.** *The Aggregate Keyword Routing problem is NP-Hard.*

**Proof.** The classical Euclidean Traveling Salesman Problem (Euclidean TSP) can be reduced to AKR problem. Given an Euclidean TSP problem, it includes a start point  $o$ , and a set of points  $S$  which the salesman should travel to, with all the points in an Euclidean space. We can construct an AKR problem as follows. We assign each point in  $S \cup \{o\}$  unique keywords with its location unchanged. Let the query point set  $Q = \{o\}$ , the aggregate keyword set  $\kappa_a = \{\kappa(o)\}$ , and the task keyword set  $\kappa_t = \{\bigcup_{p \in S} \kappa(p)\}$ . Clearly, this AKR query solves the Euclidean TSP problem. Thus, the AKR problem is NP-Hard.  $\square$

## 4 Tree Expansion

As mentioned in Sect. 2, existing algorithm TAR finds the optimal solution of an AKR query. However, the algorithm is time-consuming since its time complexity grows exponentially as the size of input increases. On the other hand, CBA is efficient, but usually results in inaccurate answers. In this section, we propose a new algorithm, which is as efficient as CBA, but much more accurate than CBA.

We first find the aggregate point  $m$ , on the basis of the *minimum cover circle*, i.e., a circle that covers all query points and has the smallest radius. Assuming that the circle is centered at the point  $o$ , we search for the aggregate point  $m$  within the neighborhood of  $o$ , because the points near  $o$  probably have small maximum distance to the query points. Then, we look for suitable POIs during the process of generating the routes. We consider the selected aggregate point  $m$  as the root of a tree, and the query points are the leaves of the tree. Initially, the tree only contains the direct path from all the leaves to the root, without passing any of the task POI. We then expand the tree by adding suitable POIs to the paths. As **maximum** is the aggregate distance function, among the  $|Q|$  paths from  $q \in Q$  to the root, it strategically selects the shortest one to perform the expansion to avoid the case where one of the path becomes very long, until all the task keywords specified in  $\kappa_t$  have been covered. The algorithm is named as *Tree Expansion (TE)* to reflect the nature of the search.

Before we present the detailed algorithm, we first introduce a min-heap where each element of the heap is in the form of  $\langle (s, p, e), d \rangle$ . Here,  $(s, p, e)$  is a three tuple vector and  $d$  indicates the detour caused if a route from  $s$  to  $e$  needs to take

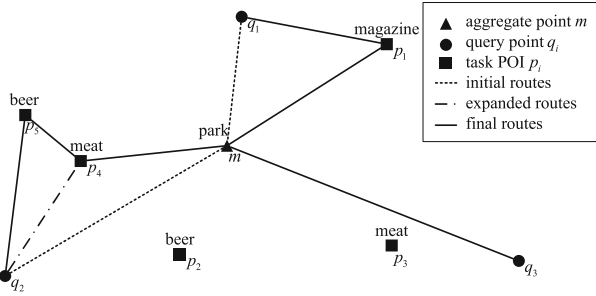


Fig. 2. Example of tree expansion

a detour at point  $p$ , i.e.,  $d = |s, p| + |p, e| - |s, e|$ . The notation  $|a, b|$  represents the Euclidean distance between two points  $a$  and  $b$ . All the elements in the heap are sorted based on ascending order of  $d$  values. We maintain  $|Q|$  min-heaps to facilitate the expansions of the paths from  $q \in Q$  to  $m$ .

Algorithm 1 lists the pseudo code of TE. It locates the minimum circle covering all the query points and uses the center of the circle to find an aggregate point  $m$  (line 1), initializes the result set  $R_{result}$  with  $R_{Q,m}^0$ , and initializes min-heap  $heap[i]$  and an index array  $leng[i]$  (lines 2–4). The index array is to record the length  $L(r_{q_i,m})$  for each route  $r_{q_i,m}$  in the current  $R_{result}$ . Thereafter, for each single query point  $q_i \in Q$ , it pushes the potential expansion options to the corresponding min-heap  $heap[i]$  to enable the tree expansion. To be more specific, for each direct route  $\langle q_i, m \rangle$ , it could be expanded by adding one POI point. We locate all the POIs points  $p \in P$  that cover at least one queried task keyword as the candidate points to enable the expansion, and add  $\langle (q_i, p, m), d \rangle$  to the heap  $heap[i]$  as a potential expansion option (lines 5–7). The real expansion is guided by routes in  $R_{result}$  and the elements in  $heap$ . Every time, we pick the path  $r_{q_i,m} \in R_{result}$  with the shortest distance for expansion, following the top option in  $heap[i]$  with the smallest detour. Assume the path  $r_{q_{index},m}$  has the shortest route distance value among all the routes in  $R_{result}$ , and the top element of  $heap[index]$  is  $e$  in the format of  $\langle (p_s, p_m, p_e), d \rangle$ . The corresponding expansion is to expand the direct link  $\langle p_s, p_e \rangle$  to  $\langle p_s, p_m, p_e \rangle$ . Note that  $p_s$  and  $p_e$  might not be adjacent because other point(s) might have been added between  $p_s$  and  $p_e$ . Meanwhile, we also check whether  $\kappa(p_m)$  is still required by  $\kappa$  with  $\kappa$  recording the task keywords not yet been covered by any route in  $R_{result}$ . We then perform the expansion if it is valid (line 12), update  $leng[index]$  to reflect the extended length of the route  $r_{q_{index},m} \in R_{result}$  and update  $\kappa$  to remove the task keywords covered by new POI  $p_m$  (line 13). In addition, the new link between  $p_s$  and  $p_m$  and that between  $p_m$  to  $p_e$  provide new expansion options. We update the heap  $heap[index]$  to reflect the new expansion options (lines 14–16). The above expansion continues until all the queried task keywords have been fully covered.

We plot one example in Fig. 2 to illustrate the search. The query points  $Q = \{q_1, q_2, q_3\}$ , the candidate aggregate point is  $m$ , and the task keywords  $\kappa_t = \{meat, beer, magazine\}$ . Initially, the tree contains only three direct routes

---

**Algorithm 1.** Tree Expansion (TE) Algorithm
 

---

**Input:**  $P, Q, \kappa_a, \kappa_t$   
**Output:**  $R_{result}$

- 1  $o := \text{getCenter}(Q), \alpha := \{p \in P | \kappa(p) \supseteq \kappa_a\}, m := \text{NearestNeighbor}(o, \alpha)$
- 2  $R_{result} := \cup_{\forall q_i \in Q} \langle q_i, m \rangle, \kappa := \kappa_t$
- 3 **for each**  $i \in |Q|$  **do**
- 4      $\text{heap}[i] := \emptyset, \text{leng}[i] := |q_i \in Q, m|$
- 5 **for each**  $p \in P$  **with**  $\kappa(p) \cap \kappa_t \neq \emptyset$  **do**
- 6     **for each**  $q_i \in Q$  **do**
- 7          $d := |q_i, p| + |p, m| - |q_i, m|$ , push element  $\langle (q_i, p, m), d \rangle$  to  $\text{heap}[i]$
- 8 **while**  $\kappa \neq \emptyset$  **do**
- 9      $\text{index} := \text{argmin}_{i \in |Q|} \text{leng}[i]$
- 10     $e \langle (p_s, p_m, p_e), d \rangle := \text{pop}(\text{heap}[\text{index}])$
- 11    **if**  $p_s$  and  $p_e$  are adjacent in a route in  $R_{result}$  and  $\kappa \cap \kappa(p_m) \neq \emptyset$  **then**
- 12      update  $r_{q_i \text{index}, m} \in R_{result}$  by including  $p_m$  in the middle of  $p_s$  and  $p_e$
- 13       $\text{leng}[\text{index}] := \text{leng}[\text{index}] + d; \kappa := \kappa - \kappa(p_m)$
- 14      **for each**  $p \in P$  **with**  $\kappa(p) \cap \kappa \neq \emptyset$  **do**
- 15          $d_1 := |p_s, p| + |p, p_m| - |p_s, p_m|$ , push  $\langle (p_s, p, p_m), d_1 \rangle$  to  $\text{heap}[\text{index}]$
- 16          $d_2 := |p_m, p| + |p, p_e| - |p_m, p_e|$ , push  $\langle (p_m, p, p_e), d_2 \rangle$  to  $\text{heap}[\text{index}]$
- 17 **return**  $R_{result}$

---

from query points to  $m$ , i.e.,  $R_{Q,m} = \{\langle q_1, m \rangle, \langle q_2, m \rangle, \langle q_3, m \rangle\}$ . At first,  $\langle q_1, m \rangle$  is the shortest route in  $R_{Q,m}$ , so we expand  $\langle q_1, m \rangle$  to  $\langle q_1, p_1, m \rangle$ . Then,  $\langle q_2, m \rangle$  becomes the shortest route, and we expand it to  $\langle q_2, p_4, m \rangle$ . After expansion,  $\langle q_2, p_4, m \rangle$  is still the shortest, and we further expand it to  $\langle q_2, p_5, p_4, m \rangle$  to complete the search.

TE only considers one aggregate point and uses this point to compute the route set. However, the aggregate point nearest to the center of the circle that covers all the query points may not be the best choice. As a result, TE may suffer from high error rate because of this not-ideal aggregate point. In order to reduce the side effect of selecting a not-ideal aggregate point on the accuracy of the approximate result, we can extend TE via evaluating multiple aggregate points. That is, we can evaluate the aggregate points according to their proximity to the center of the circle (supported by ANN algorithms). For each of the evaluated aggregate points, we generate the result routes. Parameter  $R_{candidate}$  maintains the best result route found so far. The evaluation continues until we reach an aggregate point  $m$  such that  $L_{max}(R_{Q,m}^0)$  is longer than  $L_{max}(R_{candidate})$ , or all the aggregate points have been evaluated. To differentiate from the original TE algorithm, we name the extended approximation algorithm that evaluate multiple aggregate points as TE-ext. It is noted that TE-ext is able to find a result set with higher accuracy, as compared with TE. However, it requires longer running time in most, if not all, cases. We will report the performance comparison between TE and TE-ext in the experimental study.

Following the same idea, we extend the CBA algorithm originally proposed in [1] as well, and discover that the accuracy is improved after the extension.

In the following, we use CBA-ext to represent the algorithm after extension for convenience.

## 5 Average Aggregate Distance

In Sect. 4, we presented the TE algorithm, based on **maximum** aggregate distance function. In this section, we introduce **average** as another common and useful aggregate distance function, and illustrate how TE algorithm and its extension could be adapted to different distance functions.

First, we explain the necessary changes we have to make to TE in order to support AKR queries when **average** is adopted as the aggregate distance function. We need to change the routing approach in order to have the average distance of the answer route set as small as possible. The original routing approach is to assign a new task POI to the shortest route so its impact on the maximum distance of the route set could be minimized. When considering **average**, we want to look for a route with the smallest increase in terms of its distance after adding a new POI. Hence, we only need to maintain one heap instead of  $|Q|$  heaps for supporting **average** aggregate distance function, and all the elements  $\langle (q_i, p, m), d \rangle$  in the heap are still sorted based on ascending order of the  $d$  values. When we perform the expansion, we pop out the top element from the heap as the corresponding expansion incurs the smallest detour.

Next, we explain how we adjust the extension of TE to support AKR under **average** aggregate distance function. The main idea behind the extension remains valid regardless of the aggregate distance function adopted, as checking multiple aggregate points will not bring any harm to the accuracy. However, we want to highlight that aggregate points  $m$  shall be evaluated based on  $L_{avg}(R_{Q,m}^0)$  but not  $L_{max}(R_{Q,m}^0)$ .

We also adapt TAR and CBA to support **average** aggregate distance, and the performance of all algorithms with respect to two distance functions is evaluated in the following experimental study.

## 6 Complexity Evaluation

In this section, we will analyze the worst-case time complexity of our algorithms, as well as TAR and CBA, with respect to two different aggregate distance functions.

### 6.1 Maximum Aggregate Distance Function

For our algorithm TE, it takes  $O(|P|)$  to decide  $m$  and  $O(|Q||P|\log(|P|))$  to initialize  $|Q|$  min-heap  $heap[i]$ . Then, it invokes the **while**-loop to perform the expansion, up to  $|\kappa_t|$  times. For each execution of the loop, it takes  $|Q|$  to locate the index of the route  $r_{q_i,m} \in R_{result}$  with the shortest length, and it inserts at most  $|P|$  elements to the heap. Therefore, the maximum size of the heap

is  $|\kappa_t||P|$ , and the time complexity is  $O(|\kappa_t||P| \log(|\kappa_t||P|))$ . The overall time complexity is  $O(|\kappa_t|(|Q| + |P| \log(|\kappa_t||P|)))$ .

Next, we analyze the time complexity of TAR. In Search and Assignment Phase, it takes  $O(|P|)$  to decide  $m$  and it checks at most  $|\alpha|$  aggregate points. In Task Finishing Phase, there are  $|Q|$  heaps, and most time is spent on pushing and popping heap elements. It pushes elements into heap at most  $2^{|\kappa_t|}|P|$  times, and each push involves  $|P|$  elements. Hence, the time complexity is  $O(2^{|\kappa_t|}|P|^2|\kappa_t| \log(|P|))$ . In total, the time complexity of TAR is  $O(|P| + |\alpha|(2^{|\kappa_t|}|P|^2|\kappa_t||Q| \log(|P|)))$ .

For CBA, it takes  $O(|P|)$  to locate the aggregate point  $m$ , too. The time complexity of every execution of the loop is  $O(|Q| + |\kappa_a|)$ , while it is repeated for  $|\kappa_t|$  times. Consequently, the total time complexity is  $O(|P| + |\kappa_t|(|Q| + |\kappa_t|))$ .

When we also consider extensions, we need to multiply the routing cost by  $|\alpha|$ . As the result, the time complexity of TE-ext is  $O(|\alpha||\kappa_t|(|Q| + |P| \log(|\kappa_t||P|)))$  and the time complexity of CBA-ext is  $O(|P| + |\alpha||\kappa_t|(|Q| + |\kappa_t|))$ .

## 6.2 Average Aggregate Distance Function

For TE, it only needs to use one heap during the search and the expansion is guided by the top element of the heap. Accordingly, the time complexity is changed to  $O(|\kappa_t||P| \log(|\kappa_t||P|))$ . Based on the worst-case time complexity, TE runs faster with **average** aggregate distance function.

For TAR, its time complexity remains unchanged when the aggregate distance function is changed from **maximum** to **average**. However, we want to highlight that its real performance under **maximum** is better than that under **average**, as some of its optimizations become less stronger under **average**.

For CBA, it needs to check all  $|Q|$  routes in order to find the one with the smallest detour to accommodate a new POI, so its time complexity is changed to  $O(|P| + |Q||\kappa_t|^2)$ , and it becomes slower when aggregate distance function is changed from **maximum** to **average**.

We should point out that TAR and the extension version of approximate algorithms use ANN algorithm to enumerate aggregate points. As ANN is not the focus of our paper, we skip the time complexity of the ANN algorithms.

## 7 Experimental Evaluation

In this section, we evaluate the performance of all algorithms using real datasets. In the following, we first introduce the experimental settings and then report the experimental results. All the experiments are performed on a Windows 10 machine with an Intel Core i7-4790 CPU and 32 GB memory.

### 7.1 Experimental Setup

**Dataset.** We use two real POI datasets, namely **Shanghai** and **New York**, with their main characteristics presented in Table 2. Shanghai dataset has in total

**Table 2.** Characteristics of the dataset

Dataset	# of POIs	# of keywords	# of distinct keywords
Shanghai from Amap	1,229,313	2,950,336	1,361
New York from FourSquare	132,263	249,918	711

**Table 3.** Query parameters

Parameter	Range
Query Point Number ( $ Q $ )	2, <b>4</b> , 6, 8, 10, 12
Aggregate Point Number ( $ \alpha $ )	1, 10, 100, <b>1000</b> , 10000
Task Keyword Number ( $ \kappa_t $ )	2, 4, <b>6</b> , 8, 10
Task Keyword Popularity ( $\rho_{\kappa_t}$ )	1, 10, 100, <b>1000</b> , 10000
Area Size (AS)	0.3%, 0.9%, 3%, <b>9%</b> , 30%, 90%

1, 229, 313 POIs, covering 2, 950, 336 keywords (i.e., 2.4 keywords per POI). The number of distinct keywords is 1,361, so every keyword corresponds to 903 POIs on average. New York dataset has in total 132, 263 POIs, covering 249, 918 keywords (i.e., 1.89 keywords per POI). The number of distinct keywords w.r.t. New York dataset is 711. The number of POIs corresponding to one keyword  $w_i$  could be very different from that of another keyword  $w_j$ , e.g., the number of convenience stores is much larger than the number of museums. In addition, the number of keywords associated with one POI could be also very different from that associated with another POI, e.g., a shopping mall POI could have a long list of keywords such as *shopping*, *dinning*, *bank*, *cinema*, *supermarket*, while a rail station could have only one keyword.

**Queries and Parameters.** We randomly generate queries with selected parameters to evaluate the performance of different algorithms on various settings. Each AKR query has three input parameters, the query points  $Q$ , the aggregate keyword set  $\kappa_a$ , and the task keyword set  $\kappa_t$ . Based on these three input parameters, we set five parameters to control query generation, as listed in Table 3. The values with bold numbers represent the default settings. Parameter  $|Q|$  determines the number of query points, ranging from 2 to 12; parameter  $|\alpha|$  specifies the qualified aggregate points which is dependent on  $\kappa_a$ ; parameter  $|\kappa_t|$  represents the number of queried task keywords; parameter  $\rho_{\kappa_t}$  indicates the popularity of a query task keyword with its value representing the total number of POIs in  $P$  that cover this keyword; parameter AS determines a square-shaped subarea  $S_{sub}$  within which query points are randomly generated, and it is represented as the ratio of the size of the subarea  $S_{sub}$  to that of the whole search space. For simplicity, we assume all the query task keywords in  $\kappa_t$  share the same popularity.

**Algorithms.** We implement our proposed TE and TE-ext algorithms, as well as TAR, CBA and CBA-ext, which are proposed in [1], in total 5 algorithms.

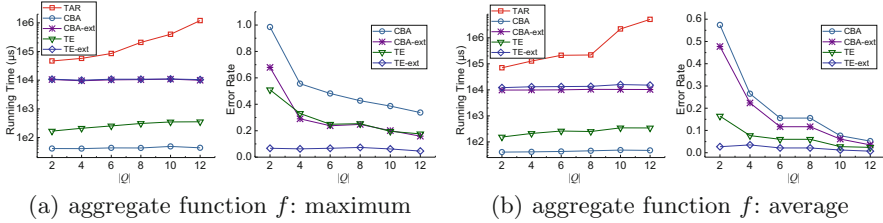


Fig. 3. Search performance vs. parameter  $|Q|$

We test their performance for both **maximum** aggregate distance function and **average** aggregate distance function.

### 7.2 Experimental Results

For every parameter setting, we randomly generate 100 queries for testing and report their average performance. All the algorithms need to form the candidate set  $\alpha$  for the aggregate points. Therefore, we exclude the cost of forming  $\alpha$  from all the experimental figures. We adopt **running time** and **error rate** as the main performance metrics with error rate set to  $\frac{L_f(R_{approx}) - L_f(R_{exact})}{L_f(R_{exact})}$ .  $R_{approx}$  refers to the result set returned by an approximate algorithm, while  $R_{exact}$  refers to the exact search result. Let the error rate be  $E$ , the accuracy is  $\frac{1}{(1+E)}$ . In other words, a lower error rate is equivalent to a higher accuracy. When we investigate the impacts of different parameters, we only report the results corresponding to Shanghai dataset, as the observations made from New York datasets are similar. We will briefly present the results of New York dataset at the end of this section.

**Impact of  $|Q|$ .** We first evaluate the impact of the size of the query point set on the performance via changing  $|Q|$  from 2 to 12. Figure 3 depicts the result. The performance gap in different algorithms is obvious and consistent across all the testing cases. TAR’s running time grows exponentially when  $|Q|$  increases, and we also observe that the running time of TAR grows even faster with **average** function than **maximum** function. On the other hand,  $|Q|$  does not change the running time of CBA, TE and their extensions much; while the increase of  $|Q|$  does help to reduce the error rate of approximate algorithms and their extended versions. This is because with more query points, the average number of POIs passed by each route from a query point to the aggregate point decreases, which makes it easier for the approximate algorithms to find a better answer. However, there is not much room for further improvement in TE-ext, since the accuracy has already been very high when  $|Q|$  is small. In general, the approximate algorithms run much faster than their extended versions but their error rates are also higher.

**Impact of  $|\alpha|$ .** Secondly, we change the parameter  $|\alpha|$  from 1 to 10,000 and report the result in Fig. 4. For TAR, CBA-ext and TE-ext,  $|\alpha|$  mainly affects the number of aggregate points enumerated from  $\alpha$ . On the other hand, both CBA and TE only evaluate one aggregate point, so  $|\alpha|$  only effects the time required

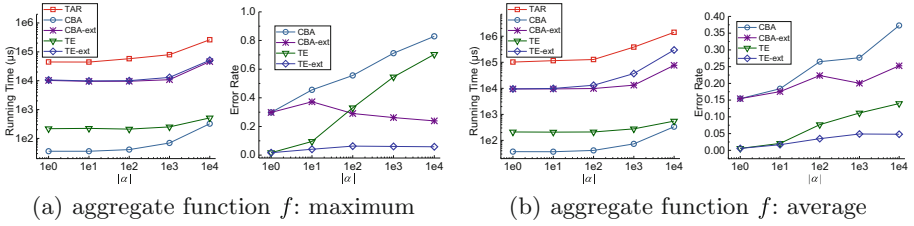


Fig. 4. Search performance vs. parameter  $|\alpha|$

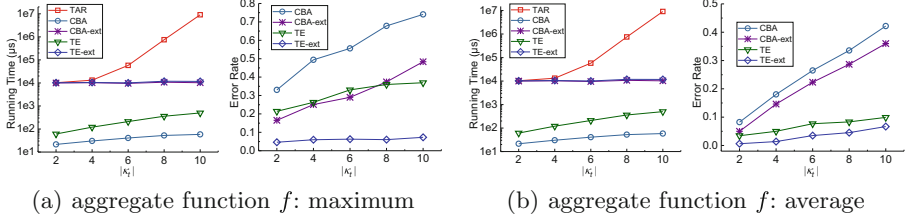


Fig. 5. Search performance vs. parameter  $|\kappa_t|$

to find the aggregate point nearest to the center of query points. We can observe that when  $|\alpha|$  values are small, an increase of  $|\alpha|$  value does not necessarily increase the running time, as many aggregate points could be filtered out. Note that we select the query points within a small subarea while the aggregate points are distributed across the entire search space. However, as  $|\alpha|$  becomes much larger (e.g.,  $|\alpha| = 1000$ ), a further increase of  $|\alpha|$  actually increases the number of aggregate points that require evaluation and hence the running time. On the other hand, unlike  $|Q|$ , the increase of  $|\alpha|$  does not decrease but increase the error rate.

**Impact of  $|\kappa_t|$ .** We now study the impact of parameter  $|\kappa_t|$ . The result is plotted in Fig. 5. Compared with  $|Q|$ ,  $|\kappa_t|$  has an even bigger impact on the performance of TAR. However, its impact on CBA and TE is less significant. This is because both algorithms only evaluate one route set, for the selected aggregate point, without enumerating all the possible route sets. As  $|\kappa_t|$  becomes bigger, the route set needs to pass by more POI points which increases the cost of generating one route. In general, TE and TE-ext are able to achieve a much lower error rate.

**Impact of  $\rho_{\kappa_t}$ .** Next, we study the impact of the popularity  $\rho_{\kappa_t}$  of query task keywords with the result plotted in Fig. 6. It is observed that  $\rho_{\kappa_t}$  value affects the performance of CBA and TE significantly. In addition, when  $\rho_{\kappa_t}$  reaches very big values, CBA and TE do not have much advantage over their extended versions in terms of running time. Similarly, their extended versions do not demonstrate much advantage over their original algorithms in terms of error rate. This is because when the density of task POIs becomes very high, there are always task POIs around a given aggregate point. In addition, we also observe that when **maximum** is selected as the aggregate distance function, the increase of  $\rho_{\kappa_t}$



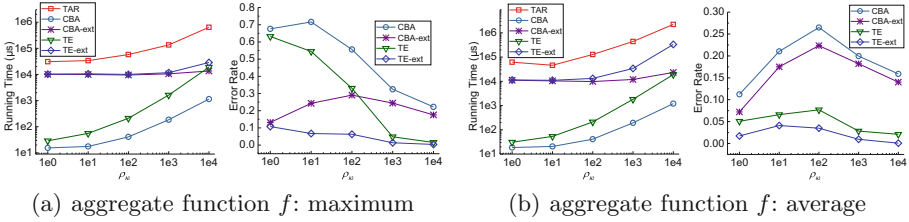


Fig. 6. Search performance vs. parameter  $\rho_{\kappa_t}$

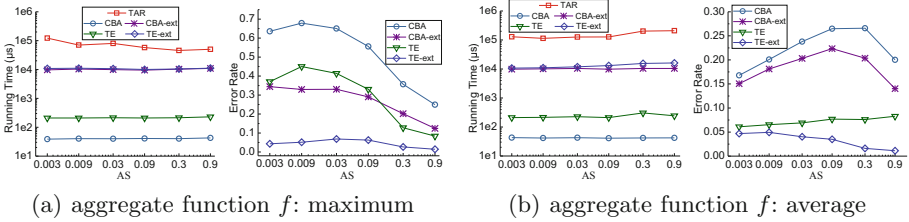


Fig. 7. Search performance vs. parameter AS

value helps to reduce the error rate. This is because as the density of task POIs increases, the position of aggregate point becomes less sensitive and the impact of a wrong aggregate point becomes smaller. However, when **average** is selected as the aggregate distance function, the error rate corresponding to very small  $\rho_{\kappa_t}$  values is low. The reason behind is that as the density of task POIs is very low, the average length of the exact route is expected to be very large which becomes less sensitive to the errors. As  $\rho_{\kappa_t}$  increase its values, the length of the result set decreases and the error rate increases. However, once  $\rho_{\kappa_t}$  reaches a large value, a further increase of its value helps to decrease the error rate.

**Impact of AS.** Last but not least, we analyze the impact of parameter AS, which reflects the distance between query points. The results are plotted in Fig. 7. We observe that the values of AS do not change the algorithms’ running time much. As for the error rate, an increase of AS value helps to improve the error rate performance when **maximum** is selected as the aggregate distance function. This is because as points are further from each other, the aggregate point that is nearest to the center of the query points is expected to have a better accuracy.

**Performance Evaluation on Different Datasets and Statistical Results.**

We test all algorithms over two different datasets, Shanghai and New York dataset. Note Shanghai dataset is around 10 times larger than New York dataset. Table 4 reports the average performance of different algorithms under two different datasets. We report the results based on four metrics, including *average running time (ART)*, *average error rate (AER)*, *maximum running time rate (MRTR)*, and *maximum error rate (MER)*. Let  $QU$  be the set of queries in our evaluation, and let  $t_i$  and  $e_i$  be the running time and error rate of each query

**Table 4.** Statistical data in Shanghai and New York

Metrics		ART (ms)		AER		MRTR		MER	
		Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.
Shanghai	TAR	590.46	3153.53	-	-	31.39	18.90	-	-
	CBA	0.29	0.30	0.50	0.22	2.91	3.03	2.32	1.31
	CBA-ext	43.96	79.17	0.26	0.18	8.26	8.96	1.91	1.00
	TE	3.09	3.57	0.28	0.07	2.99	3.21	1.71	0.74
	TE-ext	27.13	191.93	0.05	0.02	2.71	7.03	0.56	0.19
New York	TAR	34.44	181.76	-	-	51.73	32.34	-	-
	CBA	0.01	0.01	0.62	0.29	8.33	5.81	2.25	1.22
	CBA-ext	0.37	0.53	0.37	0.24	8.60	6.04	2.01	1.22
	TE	0.19	0.18	0.35	0.09	5.69	7.27	1.88	0.77
	TE-ext	0.76	3.25	0.09	0.03	6.89	5.33	1.07	0.52

$qu_i \in QU$  respectively. ART is set to  $\frac{\sum_{qu_i \in QU} t_i}{|QU|}$ ; AER is set to  $\frac{\sum_{qu_i \in QU} e_i}{|QU|}$ ; MRTR is set to  $\max_{qu_i \in QU}(\frac{t_i}{\bar{t}})$ , where  $\bar{t}$  denotes the ART of queries which have same parameters as  $qu_i$ ; and MER is set to  $\max_{qu_i \in QU}(e_i)$ . In general, the algorithms become efficient but inaccurate if maximum aggregate distance is used. TAR is 15 times slower than TE-ext, and its MRTR is high so its performance is not stable. CBA is the most efficient, but the least effective with the accuracy far below that of TE-ext algorithm. Compared with TAR, TE-ext has high accuracy (ranging from 92% to 98%) too, with respect to all two aggregate distance functions.

## 8 Conclusion

In this paper, we study the AKR problem, and propose novel approximate algorithms to process AKR queries. The algorithms support both maximum and average aggregate distance functions, two commonly used distance functions in practice. Our TE-ext algorithm is efficient since it is 15 times faster than the exact algorithm, whereas it is effective and achieves the accuracy of at least 91%. In this paper, our AKR query only returns one result, while we plan to extend the search to return top- $k$  results in the near future. In addition, we are also exploring AKR query on road networks.

**Acknowledgment.** This research is supported in part by the National Natural Science Foundation of China under grant 61772138, the National Key Research and Development Program of China under grant 2018YFB0505000, and the National Research Foundation, Prime Ministers Office, Singapore under its International Research Centres in Singapore Funding Initiative.

## References

1. Chen, K., Sun, W., Tu, C., Chen, C., Huang, Y.: Aggregate keyword routing in spatial database. In: SIGSPATIAL, pp. 430–433. ACM (2012)
2. Cong, G., Jensen, C.S., Wu, D.: Efficient retrieval of the top-k most relevant spatial web objects. *Proc. VLDB Endowment* **2**(1), 337–348 (2009)
3. De Felipe, I., Hristidis, V., Rishé, N.: Keyword search on spatial databases. In: ICDE, pp. 656–665. IEEE (2008)
4. Deng, K., Sadiq, S., Zhou, X., Xu, H., Fung, G.P.C., Lu, Y.: On group nearest group query processing. *TKDE* **24**(2), 295–308 (2012)
5. Li, F., Yao, B., Kumar, P.: Group enclosing queries. *TKDE* **23**(10), 1526–1540 (2011)
6. Li, G., Feng, J., Xu, J.: Desks: direction-aware spatial keyword search. In: ICDE, pp. 474–485. IEEE (2012)
7. Li, Z., Xu, H., Lu, Y., Qian, A.: Aggregate nearest keyword search in spatial databases. In: APWEB, pp. 15–21. IEEE (2010)
8. Li, Z., Lee, K.C., Zheng, B., Lee, W.C., Lee, D., Wang, X.: Ir-tree: an efficient index for geographic document search. *TKDE* **23**(4), 585–599 (2011)
9. Lian, X., Chen, L.: Probabilistic group nearest neighbor queries in uncertain databases. *TKDE* **20**(6), 809–824 (2008)
10. Luo, Y., Chen, H., Furuse, K., Ohbo, N.: Efficient methods in finding aggregate nearest neighbor by projection-based filtering. In: Gervasi, O., Gavrilova, M.L. (eds.) ICCSA 2007. LNCS, vol. 4707, pp. 821–833. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-74484-9\\_70](https://doi.org/10.1007/978-3-540-74484-9_70)
11. Luo, Y., Furuse, K., Chen, H., Ohbo, N.: Finding aggregate nearest neighbor efficiently without indexing. In: Proceedings of the 2nd international conference on Scalable information systems. p. 48. Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering (2007)
12. Papadias, D., Shen, Q., Tao, Y., Mouratidis, K.: Group nearest neighbor queries. In: ICDE, pp. 301–312. IEEE (2004)
13. Papadias, D., Tao, Y., Mouratidis, K., Hui, C.K.: Aggregate nearest neighbor queries in spatial databases. *TODS* **30**(2), 529–576 (2005)
14. Sharifzadeh, M., Shahabi, C.: Vor-tree: R-trees with voronoi diagrams for efficient processing of spatial nearest neighbor queries. *Proc. VLDB Endowment* **3**(1–2), 1231–1242 (2010)
15. Sun, W.W., Chen, C.N., Zhu, L., Gao, Y.J., Jing, Y.N., Li, Q.: On efficient aggregate nearest neighbor query processing in road networks. *JCST* **30**(4), 781–798 (2015)
16. Sun, W., et al.: Merged aggregate nearest neighbor query processing in road networks. In: CIKM, pp. 2243–2248. ACM (2013)
17. Sun, W., Chen, C., Zheng, B., Chen, C., Zhu, L., Liu, W., Huang, Y.: Fast optimal aggregate point search for a merged set on road networks. *Inform. Sci.* **310**, 52–68 (2015)
18. Yao, B., Tang, M., Li, F.: Multi-approximate-keyword routing in GIS data. In: SIGSPATIAL, pp. 201–210. ACM (2011)
19. Yiu, M.L., Mamoulis, N., Papadias, D.: Aggregate nearest neighbour queries in road networks. *TKDE* **17**(6), 820–833 (2005)
20. Zhang, C., Zhang, Y., Zhang, W., Lin, X.: Inverted linear quadtree: efficient top K spatial keyword search. *TKDE* **28**(7), 1706–1721 (2016)

21. Zhang, D., Chee, Y.M., Mondal, A., Tung, A.K., Kitsuregawa, M.: Keyword search in spatial databases: towards searching by document. In: ICDE, pp. 688–699. IEEE (2009)
22. Zhang, P., Lin, H., Gao, Y., Lu, D.: Aggregate keyword nearest neighbor queries on road networks. *GeoInformatica* **22**(2), 237–268 (2018)
23. Zhu, L., Jing, Y., Sun, W., Mao, D., Liu, P.: Voronoi-based aggregate nearest neighbor query processing in road networks. In: SIGSPATIAL, pp. 518–521. ACM (2010)



# Perceiving Topic Bubbles: Local Topic Detection in Spatio-Temporal Tweet Stream

Junsha Chen<sup>1,2,3</sup>, Neng Gao<sup>2,3</sup>, Cong Xue<sup>2(✉)</sup>, Chenyang Tu<sup>2,3</sup>,  
and Daren Zha<sup>2</sup>

<sup>1</sup> School of Cyber Security, University of Chinese Academy of Sciences,  
Beijing, China

<sup>2</sup> Institute of Information Engineering, Chinese Academy of Sciences,  
Beijing, China

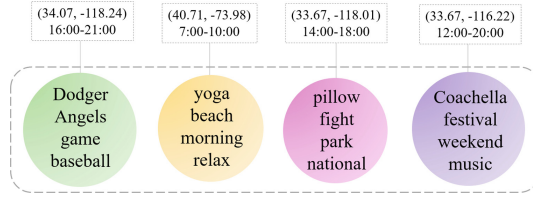
{chenjunsha, gaoneng, xuecong, tuchenyang, zhadaren}@iie.ac.cn

<sup>3</sup> State Key Laboratory of Information Security, Chinese Academy of Sciences,  
Beijing, China

**Abstract.** Local topic detection is an important task for many applications such as local event discovery, activity recommendation and emergency warning. Recent years have witnessed growing interest in leveraging spatio-temporal social media (eg. Twitter) for local topic detection. However, existing methods overlook the continuity of time and location, which is quite important and useful for local topic detection. For example, tweets posted at adjacent time and location should be considered correlated instead of isolated. To address this challenge, we propose a multi-layer heterogeneous network based embedding learner to preserve vicinity correlation as well as co-occurrence correlation, and map all the location, time, and keywords into a same latent space. Based on the heterogeneous network embedding, we develop a Bayesian mixture model to find local topics without specifying the number of topics in advance. Moreover, tweets are frequently updated, thus, we adopt an incremental update strategy to process continuous tweet stream in real time. The extensive experiments on real-world data sets demonstrate that our method outperforms the state-of-the-art existing methods.

## 1 Introduction

With the rapid development of social media like Twitter, more and more people share all kinds of topics on these platforms. Recent years have witnessed growing interest in leveraging such social media data for topic detection [12, 23]. Topics on social media range from widely-known, global ones (such as the U.S. election) to smaller-scale, local ones (such as a baseball game). The latter are crucial for various tasks such as local event discovery [21], activity recommendation [5] and emergency warning [19], among others. These local topics usually have three properties: (a) *Spatial locality*. According to First Law of Geography, when something happens in a certain place, people who are geographically close tend



**Fig. 1.** Topic bubbles with corresponding time and location. The first refers to the baseball game between Dodger and Angels; the second refers to the Yoga on beach; the third refers to the pillow fight; the fourth refers to the Coachella Festival.

to discuss the same topic. (b) *Temporal finiteness*. Local topics have certain life-cycles [18], after a certain period of time, they will fade away automatically. (c) *Semantic cohesion*. Keywords in the same topic should be semantically related, which means these words are grammatical and lexical connected [9]. We refer to topics with above three properties as **topic bubbles**, as shown in Fig. 1.

The task of topic bubble detection was extremely difficult years ago due to the lack of data sources, however, it has been made possible recently because of the proliferation of spatio-temporal social media, such as Twitter. Some studies [8, 10, 16] have already been done to leverage tweets with time and location for local topic detection. Nevertheless, there are several unique challenges that largely limit the performance of existing methods:

- (1) *Capturing vicinity correlation of spatio-temporal information*. Tweets posted at adjacent time and location are correlated instead of isolated, therefore, it is essential to capture the continuity of time and location. Existing methods, however, only take co-occurrence correlation into account but overlook vicinity correlation of spatio-temporal information [23, 24].
  - The co-occurrence correlations exist between two units when they co-occur in the same tweet. For instance, a tweet contains a spatial unit (40.57, -74.04), a temporal unit (2018-03-21 10:36:366) and some keywords (snowstorm, blizzard), whose co-occurrences reflect they all have similar semantics.
  - The vicinity correlations are derived from the continuity of both location and time. For example, when a *Tax Protest Rally* happens at Time Square in the afternoon, tweets posted around Time Square and during that afternoon should be considered correlated instead of isolated.
- (2) *Generating the number of topic bubbles automatically*. To ensure high quality of topic bubbles, it's vital to generate the number of topics automatically. However, due to the lack of prior knowledge, many existing methods need to manually specify the number of topics in advance [2, 22], which causes the topic bubbles to be less accurate and pragmatic.
- (3) *Processing continuous tweet stream in real time*. Tweet stream is a set of continuous data generated in real time. Therefore, it is desirable to continuously process the massive tweet stream to allow for timely actions. Such a requirement renders existing batch-wise methods [11, 20] inapplicable.

To address above challenges, we propose a heterogeneous network embedding-based method underpinned by two major modules, to detect topic bubbles in spatio-temporal tweet stream. The first module is a multi-layer heterogeneous network based embedding learner which jointly maps all the location, time, and keywords into the same latent space with both their co-occurrence and vicinity correlations preserved. If two units are highly correlated (eg. *protest* and *against*, 5th Ave region and the keyword *shopping*, or 8am and the keyword *traffic-jam*), their representations in the latent space tend to be close. The heterogeneous network based embedding module can not only capture the subtle semantic similarities among location, time and keywords, but also maintain the correlations of adjacent location and continuous time. Based on the embedding results, the second module is Bayesian mixture clustering which aims at finding high-quality topic bubbles without specifying the number of topics in advance. To be specific, we develop a novel Bayesian mixture model that can divide the keywords of tweets into a number of topic bubbles. The model uses von-Mise Fisher (vMF) distribution over the embedding results of words. Furthermore, as the query window shifts continuously, we employ an incremental updating strategy to ensure excellent efficiency, which means our method does not need to detect topic bubbles in the new query window from scratch, but just needs to update the previous results with little cost to ensure fast real-time detection.

Our main contributions are summarized as follows:

- (1) We design a multi-layer heterogeneous network embedding learner that jointly maps the location, time and keywords into the same latent space with their co-occurrence and vicinity correlations preserved.
- (2) We propose a Bayesian mixture clustering model to detect topic bubbles efficiently based on the embeddings, which can find the number of topic bubbles automatically.
- (3) We adopt an incremental update strategy in our method, so that our method can process continuous tweet stream in real time with little cost when the query window shifts.

We evaluated our method on two real-world tweet data sets, each containing millions of spatio-temporal tweets. The extensive experiments show that our proposed method outperforms the existing methods significantly. Meanwhile, our method is efficient to be deployed for processing large tweet streams in practice.

## 2 Related Work

**Representation Learning.** Representation learning is a vectorization technique designed to represent an object as a low-dimensional vector. This technology solves the problem of data sparseness effectively, and improves the performance of knowledge acquisition as well as knowledge fusion. Word2vec [14] is one of the most influential methods created by Tomas Mikolov et al., reconstructing linguistic contexts of words. Meanwhile, some work about network embedding

has emerged in the past few years, such as DeepWalk [15] and node2vec [7]. However, these methods can only maintain the structure information of the network. Some recent efforts have been devoted to leveraging additional information for a better performance. SANRL [3] is a scalable joint NRL framework which preserve both structural and attributed information in the unified representations which not only learns representations of nodes, but also learns representations of attributes in the same low dimensional vector space.

**Topic Modeling.** Conventional topic models such as LDA [2] are designed to implicitly capture word co-occurrence patterns at document-level to reveal topic structures. However, these conventional topic models suffer a lot from the data sparsity problem in short texts, like tweets. Therefore, some methods are proposed specifically for modeling short text, such as BTM [22] and Twitter-LDA [25]. Recently, some work has been done to combine topic modeling with representation learning. Das et al. [4] propose a topic model which uses a Gaussian distribution over word embeddings. By performing inference over the vector representations of the words, their model is encouraged to group words that are semantically similar, leading to more coherent topics. Besides, as the emergence of the documents with spatial and temporal information, some work has been done in the area of spatio-temporal topic modeling. Sizov et al. [16] extend LDA by assuming each latent topic has a multinomial distribution over text and two Gaussians over latitudes and longitudes. Kawamae [10] propose a temporal topic model, assuming the words of a document are drawn from the user specific topic distribution and the timestamps are drawn from that topic. All the above work models either consider time or location, while LTM [12] proposed by Liu et al. takes both of them into account, but it can only detect topics in a city level.

**Spatio-Temporal Social Media Mining.** Lately, the appearance of spatio-temporal social media data has enabled progresses in all kinds of tasks besides topic modeling, such as event detection, activity recommendation and so on. Liu et al. present a probabilistic graphical model [13] using both spatial and temporal information to recommend location. Zhang et al. propose an embedding-based method [23] for spatio-temporal event detection by capturing cross-model correlations of spatio-temporal information. Nevertheless, this method only considers the co-occurrence correlations, and the vicinity correlations of time and location are not taken into consideration. Zhang et al. also propose a network-based method [24] for activity recommendation by jointly modeling all spatial, temporal and textual units into the same latent space. However, the network constructed by this method is too complicated, and different types of information are not distinguished from each other, so there exist certain limitations on modeling semantic information and processing a large amount of data.



### 3 Preliminaries

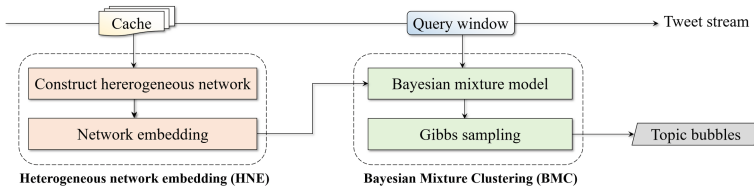
#### 3.1 Problem Description

**Definition 1 (Topic bubble).** *Topic bubble is defined as a set of keywords  $\{w_1, w_2, \dots, w_n\}$ , which satisfies the three properties: (a) spatial locality, (b) temporal finiteness and (c) semantic cohesion.*

Let  $TW$  be a corpus of tweets. Each tweet  $d \in TW$  is represented by a tuple  $\langle t_d, l_d, W_d \rangle$ , where  $t_d$  is its post time,  $l_d$  is its geo-coordinate represented by  $(x_d, y_d)$ , and  $W_d$  is a series of keywords of the tweet. Sort all tweet tuples in chronological order as tweet stream  $\{d_1, d_2, d_3, \dots\}$ .  $Q$  is the query window defined as  $[t_s, t_e]$ , where  $t_s$  and  $t_e$  are the start and end timestamps, satisfying  $t_{d_1} \leq t_s < t_e \leq t_{d_n}$ . Our method aims at detecting all topic bubbles that occur in the query window  $Q$  and updating the topic bubbles as  $Q$  shifts continuously.

#### 3.2 Framework of Our Method

The emergence of a topic often leads to a large amount of related tweets posted around its occurring location and during a certain period of time. For example, suppose a *Tax Protest Rally* happens at Time Square in New York City, many participants at Time Square will post tweets to share information during that time, using the keywords such as *protest*, *against*, *government* and *tax*. These keywords can form a keyword cluster as a topic bubble describing the protest event, since such a keyword cluster satisfies three properties mentioned before.



**Fig. 2.** The framework of our method

On the foundation of above definitions and observations, we present a topic bubble detection method with two major modules from spatio-temporal tweet stream, shown in Fig. 2. The first module is a multi-layer heterogeneous network embedding learner which can map all the location, time and keywords into a same latent space. The second module is Bayesian mixture clustering to find topic bubbles in the query window  $Q$  based on word embeddings without specifying the number of topic bubbles in advance, and adopt Gibbs sampling to estimate the parameters. Finally, by calculating the average embedding vector for each topic bubble and then performing similarity query, we can obtain corresponding time and location of each topic bubble. Furthermore, as the query window shifts

continuously, our method does not need to detect topic bubbles in the new query window from scratch, but just needs to update the previous results with little cost to ensure high efficiency.

## 4 Heterogeneous Network Embedding

### 4.1 Construct Heterogeneous Network

As shown in Fig. 3, we extract time, location coordinate and text message from tweet stream to construct a multi-layer heterogeneous network (MHN) defined as  $G = (TN, SN, KN, TL, SL, KL)$ . Text message can be easily segmented, since the keywords are natural textual units for embedding. However, post time and location coordinate are continuous variables and there are no natural embedding units. To address this issue, we break the geographical space into equal-size regions (300 m \* 300 m) and consider each region as a spatial unit. Similarly, we break one day into 24 h and consider each hour as a basic temporal unit. The explanations of all the elements in MHN are described as follows:

- $TN$  is a set of temporal units, each representing a time unit.
- $SN$  is a set of spatial units, each representing a certain region.
- $KN$  is a set of text units, each representing a keyword.
- $TL$  is a set of time-keyword links, each representing an co-occurrence correlation between a temporal unit and a text unit.
- $SL$  is a set of location-keyword links, each representing an co-occurrence correlation between a spatial unit and a text unit.
- $KL$  is a set of keyword links, each representing a co-occurrence correlation between two keywords.

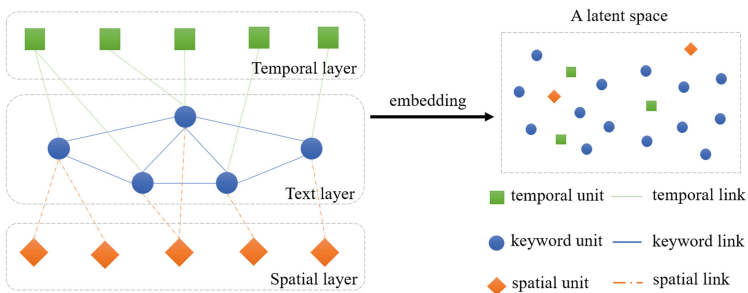


Fig. 3. Heterogeneous network embedding

All the links are undirected and weighted. The weight of links is defined as co-occurrence times between two units.

### 4.2 Network Embedding

Build upon the multi-layer heterogeneous network, we present a novel way to generate the low-dimensional vector for each unit, and preserve both co-occurrence and vicinity correlations.

**Loss Function for Textual Information.** Inspired by the network embedding algorithm [3], our method learns the representations of three types of units based on the assumption that units which have similar neighborhoods will have similar representations. Here, neighborhoods refer to both direct neighbors corresponding to first-order estimation and higher-order neighbors corresponding to higher-order estimation [7].

Consider text layer separately. Given a keyword unit  $v_k$ , we adopt weighted random walk to generate many fixed-length keyword sequences. Repeat above process for every keyword unit  $v_k \in KN$ , we can get a set of keyword unit sequences as training data  $S_{text}$ . By feeding shuffled  $S_{text}$  to skip-gram [14], which is a scalable and efficient way to learn word representations, the following objective loss function should be minimized:

$$\begin{aligned}
 O_1 &= -\log P(S_{text}) = -\sum_{v_k \in KN} \sum_{1 \leq k \leq n} \log P(S_k^{v_k}) \\
 &= -\sum_{v_k \in KN} \sum_{1 \leq k \leq n} \sum_{v_k^i \in S_k^{v_k}} \log P(v_k^{i-\frac{w}{2}}, \dots, v_k^{i-1}, v_k^i, v_k^{i+1}, \dots, v_k^{i+\frac{w}{2}}) \\
 &= -\sum_{v_k \in KN} \sum_{1 \leq k \leq n} \sum_{v_k^i \in S_k^{v_k}} \sum_{i-\frac{w}{2} \leq j \leq i+\frac{w}{2}, j \neq i} \log P(v_j | v_i)
 \end{aligned} \tag{1}$$

Where  $S_k^{v_k} = \{v_k^1, v_k^2, \dots, v_k^m\}$ ,  $n$  is the fixed-length of sequence, and  $w$  is the size of the sliding window.

**Loss Function for Temporal Information.** Consider a keyword unit co-occurring at a certain time point, take the adjacent time points as a set  $T(v_k)$ . We give the co-occurrence and vicinity patterns directly to model  $P(T(v_k))$ :

$$\log P(T(v_k)) = \sum_{t_i \in T(v_k)} \{ \log P(t_i | v_k) + \log P(v_k | t_i) + \sum_{t_j \in T(v_k), j \neq i} \log P(t_j | t_i) \} \tag{2}$$

The above three kinds of conditional probability can capture different similarities between temporal layer and text layer. By maximizing  $P(t_i | v_k)$ ,  $P(v_k | t_i)$  and  $P(t_j | t_i)$ , the co-occurrence and vicinity correlations will be well preserved, which means keywords co-occurring at adjacent time points tend to have close representations and the adjacent time points are also inclined to have close representations. Using  $S_{text.time}$  to represent the time-keyword sequences acquired in the preceding procedure, the objective loss function between temporal layer and text layer is defined as follows:

$$\begin{aligned}
O_2 &= -\log(S_{text-time}) = -\sum_{v_k \in KN} \sum_{1 \leq k \leq n} \log P(T(v_k)) \\
&= -\sum_{v_k \in KN} \sum_{1 \leq k \leq n} \sum_{t_i \in T(v_k)} \{\log P(t_i|v_k) + \log P(v_k|t_i)\} + \sum_{t_j \in T(v_k, j \neq i)} \log P(t_j|t_i)
\end{aligned} \tag{3}$$

**Loss Function for Spatial Information.** We define loss function for spatial information by analogy with loss function for temporal information. Consider a keyword unit  $v_k$  co-occurring at a location point, take the adjacent location points as a set  $L(v_k)$ , we also use the co-occurrence and vicinity patterns to model  $P(L(v_k))$ . The definition of loss function for spatial information is given directly as follows:

$$\begin{aligned}
O_3 &= -\log(S_{text-location}) = -\sum_{v_k \in KN} \sum_{1 \leq k \leq n} \log P(L(v_k)) \\
&= -\sum_{v_k \in KN} \sum_{1 \leq k \leq n} \sum_{l_i \in L(v_k)} \{\log P(l_i|v_k) + \log P(v_k|l_i)\} + \sum_{l_j \in T(v_k, j \neq i)} \log P(l_j|l_i)
\end{aligned} \tag{4}$$

**Loss Function.** To integrate above three types of loss functions into a joint representation learning framework, we adopt a weighted linear combination of  $O_1$ ,  $O_2$  and  $O_3$  to formulate our final objective loss function of the heterogeneous network embedding model:

$$O_{joint} = O_1 + aO_2 + bO_3 \tag{5}$$

Where  $a$  and  $b$  are trade-off parameters to balance three different loss functions. For example, if we focus on temporal similarity between keywords,  $a$  should be a big number and  $b$  should be a small number; if we concentrate on spatial similarity between keywords, a big  $a$  and a small  $b$  are suitable. By minimizing  $O_{joint}$ , we can get resulting vectors  $u_v$  and  $u'_v$  for each unit  $v$ , Finally,  $u_v$  of keywords will be used in the subsequent clustering module.

### 4.3 Incremental Updating

When the query window shifts, it is undesirable to re-construct the multi-layer heterogeneous network in the new query window from scratch. Therefore, we employ an incremental updating strategy to construct the multi-layer heterogeneous network in the new query window. Assume the query window shifts from  $Q$  to  $Q'$ , we use  $TW_- = \{d_1 \cdots, d_k\}$  to represent the outdated tweets, and use  $TW_+ = \{d_m, \cdots, d_n\}$  to represent the new tweets. Instead of re-constructing the multi-layer heterogeneous network for all the tweets in  $Q'$ , we simply remove the units in  $TW_-$  from the MHN and insert the units in  $TW_+$  into the MHN.

In network embedding part, we delete the sequences generated by random walk for old units and add new sequences of new units, and only initialize the new units instead of all the units to accelerate network training process.

### 4.4 Complexity Analysis

In this subsection, we analyze the computational complexity of the heterogeneous network embedding module. Our module uses hierarchical softmax technique to speed up training process. Given an arbitrary training instance whose form is an input-output pair, the computational complexity of computing  $P(output|input)$  can be reduced to  $O(\log(|TN| + |SN| + |KN|))$ .

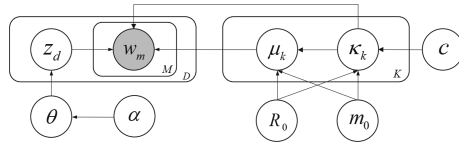


Fig. 4. Bayesian mixture model

## 5 Bayesian Mixture Clustering

Based on the heterogeneous network embedding, we develop a Bayesian mixture clustering model to divide the keywords of tweets in the query window  $Q$  into a number of clusters. Unlike traditional media, tweets are quite short, which contain a limited number of words. Therefore, we assume that each tweet is represented by only one topic rather than a mixture of topics like the standard LDA. The key idea behind our model is that every keyword cluster implies a coherent topic around a certain place and during a certain period of time. The notations used in this section are summarized as follows:

- $X$ : the set of embeddings for all the words in  $D$
- $Z$ : the set of cluster memberships for all the tweets in  $D$
- $\kappa$ : the set of  $\kappa$  for all the clusters
- $\kappa^{-k}$ : the subset of  $\kappa$  excluding the one for cluster  $k$
- $Z^{-d}$ : the subset of  $Z$  excluding tweet  $d$
- $w^k$ : the sum of the embeddings in cluster  $k$
- $w^{k,-m}$ : the sum of the word embeddings in cluster  $k$  excluding word  $m$
- $n^k$ : the number of tweets in cluster  $k$
- $n^{k,-d}$ : the number of tweets in cluster  $k$  excluding tweet  $d$

We assume that there are at most  $K$  clusters in the query window  $Q$ . At the end of the clustering process, some of these  $K$  clusters may become empty. As a result, the appropriate number of clusters in any query window can be automatically discovered. Figure 4 shows the generative process for all the words of

tweets in the query window  $Q$ . Here,  $D$  is the set of tweets in the query window  $Q$ , and  $w_m$  is the  $d$ -dimensional embedding of word  $w$  generated from the heterogeneous network embedding learner. As shown, we first draw a multinomial distribution  $\theta$  from a Dirichlet prior  $Dirichlet(\cdot|\alpha)$ ; and for modeling the word embeddings, we draw  $K$  von Mises-Fisher (vMF) distributions from its conjugate prior  $\Phi(\mu, \kappa|m_0, R_0, c)$ . Our choice of the vMF distribution is motivated by the effectiveness of the cosine similarity in quantifying the similarities between embedding vectors [23]. For each tweet  $d$ , we first draw its cluster membership  $z_d$  from  $\theta$ . Once the cluster membership is determined, we draw its embeddings of the words in this tweet from its corresponding vMF distribution.

To summarize the above generative process, we have:

$$\begin{aligned} \theta &\sim Dirichlet(\cdot|\alpha) & \{\mu_k, \kappa_k\} &\sim \Phi(\cdot|m_0, R_0, c) \\ z_d &\sim Categorical(\theta) & w_m &\sim vMF(\cdot|\mu_{z_d}, \kappa_{z_d}) \end{aligned} \quad (6)$$

After obtaining the topic bubbles, we calculate the average embedding vector for each topic bubble, and then perform similarity query to get corresponding time and location for each topic bubble.

**Gibbs Sampling.** We introduce Gibbs sampling to estimate the parameters. The key of Gibbs sampling is to obtain the posterior distribution for  $z_d$ . Due to the space limitation, we directly give the conditional probabilities for  $z_d$ :

$$\begin{aligned} p(z_d = k|X, Z^{-d}, \alpha, m_0, R_0, c) &\propto p(z_d = k|Z^{-d}, \alpha) \cdot \\ &\prod_{w_m \in d} p(w_m|X^{-m}, Z^{-d}, z_d = k, \alpha, m_0, R_0, c) \end{aligned} \quad (7)$$

The two quantities in Eq. 7 are given by:

$$p(z_d = k|\cdot) \propto (n^{k, -d} + \alpha) \quad p(w_m|\cdot) \propto \frac{C_D(\kappa_k)C_D(\|\kappa_k R_0 m_0 + w^{k, -m}\|_2)}{C_D(\|\kappa_k R_0 m_0 + w^{k, -m} + w^m\|_2)} \quad (8)$$

**Incremental Updating.** We also use the incremental updating strategy to ensure efficiency in the clustering module. Instead of performing Gibbs sampling for all the tweets in  $Q'$ , we simply drop the tweets in  $TW_-$  and sample the cluster memberships for the tweets in  $TW_+$ . Such an incremental updating strategy [23] achieves excellent efficiency and yields high-quality topic bubbles in practice.

**Complexity Analysis.** We further analyze the time complexity of Bayesian mixture clustering (BMC) module. For each iteration, BMC needs to re-sample a cluster for the  $D$  tweets in turn. For each tweet  $d$ , it computes the conditional probability  $p(z_d = k|\cdot)$  for each cluster  $k$ . The complexity of this conditional probability is linear to the average length of documents,  $\bar{L}$ . So the time complexity for each iteration is  $O(KD\bar{L})$ .

## 6 Experiment

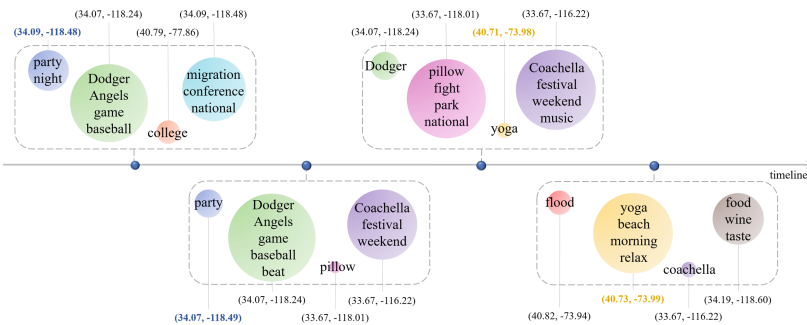
### 6.1 Experiment Settings

**Data Sets.** Our experiments are based on two real-world tweet data sets, both are crawled using Twitter Streaming API from 2018.03.21 to 2018.04.27. The first data set, referred to as NY, consists of 1.7 million spatio-temporal tweets in New York. The second data set, referred to as LA, consists of 1.9 million spatio-temporal tweets in Los Angeles.

**Baselines.** We compare our method with four existing topic detection methods which are often used in Twitter. For simplicity, we use HNE to represent the heterogeneous network embedding learner of our method, and denote the Bayesian mixture clustering module of our method by BMC.

- LDA [2] is a classic generative statistical topic model. It can find the top K keywords of each topic based on the number of topics and prior distributions.
- BTM [22] is further improved on LDA model. It learns topics by directly modeling the generation of word co-occurrence patterns in the corpus.
- Twitter-LDA [25] is a topic model proposed specifically for Twitter. It addresses the noisy nature of tweets, since it captures background words in tweets.
- LTM [12] is a spatio-temporal topic model proposed for local topic detection. It captures both spatial and temporal aspects in a probabilistic model.

Besides, in order to demonstrate the effectiveness of HNE module, we replace it with Multi-Modal Embedding (MME) presented in trioveevent [23] and Cross-Modal Embedding (CME) proposed by Zhang et al. [24]. After obtaining the embedding results from above three methods, we employ BMC to get final topic bubbles. Besides, we also replace the BMC module with GLDA (Gaussian LDA) [4] and sHDP (spherical HDP) [1] to demonstrate the effectiveness of our clustering module. Furthermore, we use a simplified version HNE- based on

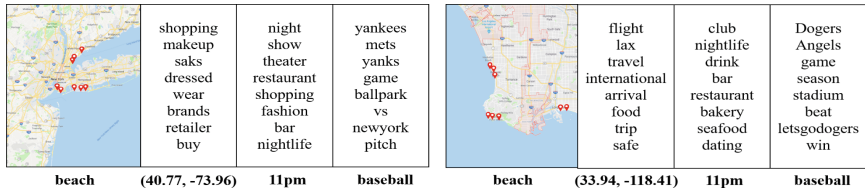


**Fig. 5.** An illustrative case of topic bubble detection. Detecting topic bubbles on two real-world data sets by shifting a 7-day query window from 2018.03.26 to 2018.04.16.

HNE to illustrate the importance of maintaining the vicinity correlation. HNE-ignores the vicinity correlation of time and location continuities while processing heterogeneous network embedding.

### 6.2 Illustrative Cases

Before comparing our method with baselines, we present an illustrative case in Fig. 5. As shown, the topic bubbles generated by our method are of high quality, and we can get an obvious topic from the keywords in each topic bubble, such as conferences, big parties and some festivals. After a period of time, existing topic bubbles will fade away while new topic bubbles will appear gradually; and the location where some topics occurring will also have a certain degree of shifts over time, like the location coordinate marked in bold in the figure. To further understand why our method is capable of generating high-quality topic bubbles, we perform similarity queries based on the embedding results of three types of units in New York and Los Angeles. As shown in Fig. 6, given the query unit, we list the top eight most similar units by computing the cosine similarity.



(a) Examples on NY (the second query is the location of the Fifth Avenue). (b) Examples on LA (the second query is the location of the LAX Airport).

**Fig. 6.** Similarity queries based on the multi-layer heterogeneous network embeddings, which are learned from the spatio-temporal tweets in New York City and Los Angeles. In each city, the first query retrieves regions relevant to the keyword ‘beach’; the second retrieves keywords relevant to a certain location; the third retrieves keywords related to a time unit ‘11pm’ and the last two retrieve relevant keywords for the given query keywords. For each query, we use the learned embeddings to compute the cosine similarities between different units, and retrieve the top eight most similar units.

### 6.3 Quantitative Results

**Effectiveness Comparison.** A good topic model will generate coherent topics. Therefore, we calculate topic coherence for the topics generated by each method respectively. Topic coherence [17] is a measure used to evaluate coherent degree of topics, which is formally defined as:

$$C = \frac{2}{N \cdot (N - 1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^N PMI(w_i, w_j) \quad PMI(w_i, w_j) = \log \frac{P(w_i, w_j) + \epsilon}{P(w_i) \cdot P(w_j)} \tag{9}$$



Data sets	LA			NY		
	1	3	7	1	3	7
LDA	0.363	0.382	0.478	0.351	0.416	0.533
BTM	0.521	0.608	0.692	0.394	0.569	0.613
Twitter-LDA	0.513	0.632	0.679	0.461	0.547	0.621
LTM	0.768	1.125	1.488	0.823	1.336	1.568
MME+BMC	0.812	1.264	1.653	0.732	1.165	1.793
CME+BMC	0.896	1.486	1.967	0.906	1.562	1.874
HNE+GLDA	0.698	0.824	1.012	0.657	0.786	0.982
HNE+sHDP	0.732	0.895	1.254	0.764	0.906	1.365
HNE→BMC	0.879	1.457	1.837	0.864	1.498	1.798
<b>HNE+BMC (our method)</b>	<b>1.163</b>	<b>1.782</b>	<b>2.135</b>	<b>1.174</b>	<b>1.809</b>	<b>2.253</b>

Fig. 7. Average topic coherence of different query window size on two data sets

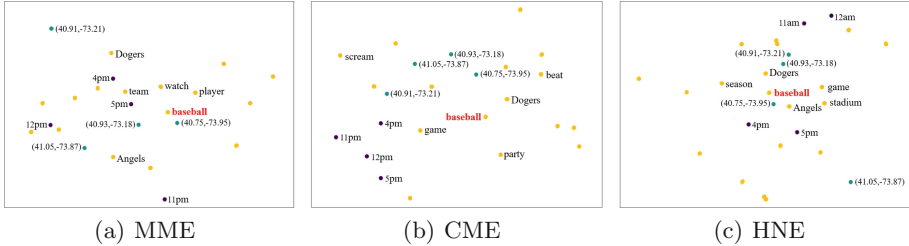
Figure 7 shows the average topic coherence of different methods in different query window size on two tweet data sets. We observe that our method outperforms the baseline methods on both data sets. Figure 8 lists the results of top-3 topics sorted by topic coherence after performing above five methods to detect the topics that took place in New York from 2018.04.16 to 2018.04.18.

It can be seen that the topics generated by LDA, BTM and twitter-LDA can not get obvious topics with time and location, because above three methods are all keyword-based methods, without using the spatio-temporal information. Thus, they fall short in detecting topic bubbles. As for LTM, it incorporates the spatio-temporal information into its model. Compared to above methods, the topic coherence has been significantly improved, and we can also obtain obvious topics from the keyword clusters with time and location. However, due to the large granularity when processing spatio-temporal information and the overlook of the vicinity correlation, the resulting topics can not reflect relatively accurate time and location. On the contrary, our method can obtain topic bubbles with

Methods	topics		spatio-temporal info	coherence
	keywords	summary		
LDA	1: show, love, browns, traffic, baker	--	--	0.615
	2: place, expressway, parkway, east, place	--	--	0.529
	3: united, states, Get, Trump, Newyork	--	--	0.387
BTM	1: incident, traffic, crush, exit, clear	traffic incident	--	0.872
	2: rain, fall, photo, think, birthday	--	--	0.669
	3: nyc, Brooklyn, stadium, block, morning	--	--	0.525
Twitter-LDA	1: rain, hard, heavy, fall, sudden	heavy rain	--	1.064
	2: game, go, career, team, mets	--	--	0.972
	3: bridge, east, cloudy, flush, downtown	--	--	0.837
LTM	1: bacon, party, beach, beer, barbacon	bacon beach party	Manhattan, 09:00-22:00	1.734
	2: wine, tour, winemakers, buffer, drink	Wine Grand Tour	Brooklyn, 13:00-22:00	1.269
	3: realmadrid, madrid, Bayern, youtu, live	Champions League	Manhattan, 10:00-18:00	1.023
<b>our method</b>	<b>1: subways, rain, swamped, drench, heavy</b>	<b>Subways swamp</b>	<b>(40.82, -73.94), 13:00-16:00</b>	<b>2.766</b>
	<b>2: mets, brewers, winning, beat, game</b>	<b>Mets vs Brewers</b>	<b>(40.81, -74.07), 16:00-21:00</b>	<b>2.145</b>
	<b>3: YAGP, stars, gala, theatre, center</b>	<b>YAGP 2018 Gala</b>	<b>(40.75, -73.93), 9:00-11:00</b>	<b>1.793</b>

Fig. 8. Top 3 topics generated by different methods in New York from 4.16 to 4.18

exact time and location, because our method handles time and location more granularly. And it can capture the semantic correlations between the keywords as well as maintain correlations with continuous time and location, leading to more coherent topics. However, when ignoring the vicinity correlation, the results of our method decrease a lot as shown in Fig. 7, which demonstrates the significance of capturing the vicinity correlation of spatio-temporal information.



**Fig. 9.** Embeddings generated by three different models given the query word *baseball*

Next, we analyze the effectiveness of two major modules in our method.

- (1) *HNE module analysis.* Given the query word *baseball*, Fig. 9 shows the most similar embedding results from three different methods. Compared to MME and CME, HNE can get better embedding results. Because MME only captures the co-occurrence relationship among location, time and keywords, but ignores the vicinity correlation of spatio-temporal continuity, resulting in the embedding results of adjacent time and location less compactly. In Fig. 9(a), for example, *11pm* and *12pm* are close in temporal dimension, but their embedding results are far from each other in the latent space. The same problem occurs in embedding results of spatial dimensions, such as  $(40.93, -73.18)$  and  $(40.91, -73.21)$ . As for CME, it's a further optimization based on MME. It also captures the relationship among location, time and keywords by constructing a heterogeneous network. However, this model does not differentiate different types of units and diminishes the semantic correlation between keywords. Thus, the keyword semantics can not be well captured. As we can see in Fig. 9(b), the time and location units tend to gather together, and it is relatively sparse between keywords. Clustering these keywords often leads to semantically irrelevant keywords in the same cluster. HNE of our method maintains both the co-occurrence correlation and vicinity correlation by stratified random walk sampling and cross-layer random walk sampling. As we can see in Fig. 9(c), the embedding results of continue time and adjacent location are close to each other, like *11am* and *12am*,  $(40.91, -73.21)$  and  $(40.93, -73.18)$ . And discontinuous time and nonadjacent location are far away from each other, such as *4pm* and *12am*,  $(40.91, -73.21)$  and  $(41.05, -73.87)$ . In addition, the embedding results of keywords, which are semantically similar and spatio-temporally close, tend to be compact in the latent space.

(2) *BMC module analysis.* As shown in Fig. 7, based on the same embedding results of keywords generated from HNE module, BMC can obtain more coherent topics than GLDA and sHDP. GLDA assumes that word vectors are generated from Gaussian distribution, and sHDP presumes that word vectors are generated from Von Mises-Fisher distribution. The superiority of the vMF distribution over other alternatives for modeling textual embeddings has also been demonstrated in recent studies on clustering [6] and topic modeling [1]. BMC is a further optimization on sHDP, specifically for short text modeling. Unlike sHDP, we assume that each tweet is represented by only one topic rather than a mixture of topics. Therefore, BMC has obvious advantages compared to sHDP when dealing with short text like tweets and processing a large amount of data.

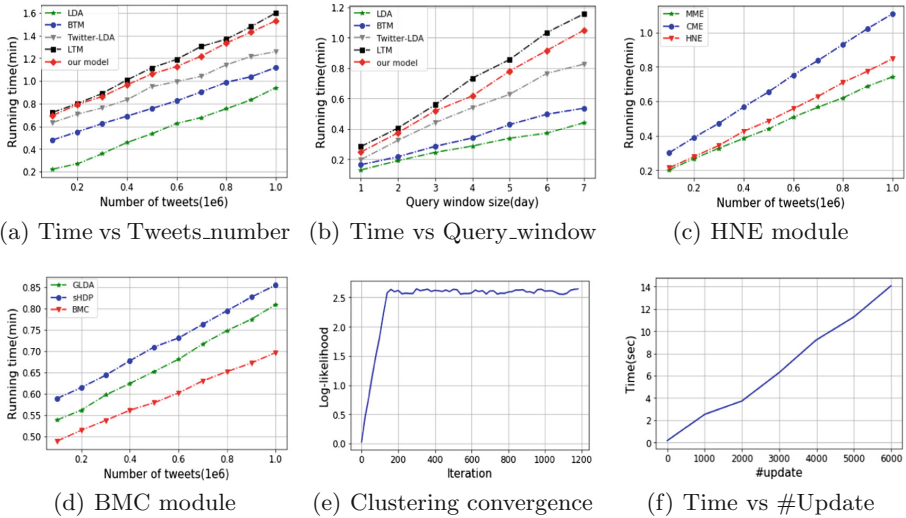


Fig. 10. Efficiency study.

**Efficiency Comparison.** We proceed to analyze the efficiency of different methods. Figure 10(a) and (b) show the running time variation of the five methods as the number of tweets and the query window size increases. We observe that our method is a little bit slower than LDA, BTM and twitter-LDA, because in addition to keywords, we also add spatio-temporal information into our method, which brings extra time cost. Compared to LTM, our method is more efficient. Although our method get topic bubbles by two modules, HNE module can obtain word vectors very quickly, and BMC module only need to sample the embedding vectors for every word. On the contrary, LTM needs to sample all the three types of information, it's time-consuming.

Next, we analyze the efficiency of two major modules in our method. As we can see from Fig. 10(c) and (d), in HNE module, the efficiency of HNE is slightly lower than MME, but significantly higher than CME. Because MME ignores the continuity of time and location, and only considers the co-occurrence correlation. As for CME, the network construction is too complicated. Thus, the efficiency is too low when processing a large amount of data. In BMC module, the efficiency of BMC is much higher than both GLDA and sHDP, because BMC specifically models short texts and assumes each tweet is represented by one topic, which simplifies the sampling procedure.

Figure 10(e) shows the log-likelihood changing as the number of Gibbs sampling iterations increases. We observe that the log-likelihood quickly converges after a few iterations. Therefore, it is usually sufficient to set the number of iterations to a relatively small value in practice for better efficiency. In Fig. 10(f), we report the scalability of our method when the number of updates (the sum of removed tweets and inserted tweets) varies. As shown, the running time is linearly increasing as the number of updates changing. The results demonstrate that our method has good scalability.

## 7 Conclusion

In this paper, we propose a novel method to detect topic bubbles in continuous spatio-temporal tweet stream. With the multi-layer heterogeneous network embedding learner which incorporates time and location into modeling, the keyword embeddings can contain semantic information as well as spatio-temporal information. Based on the keyword embeddings, we present a Bayesian mixture model to obtain topic bubbles by clustering keywords into groups. The keywords in the same group are temporally compact, geographically coherent and semantically close. Experiments show that our method can improve the effectiveness of the existing methods significantly and achieving good efficiency. For future work, we will further take other features of tweets (eg. hashtags) into consideration and evaluate our method on more real-world tweet data sets from different regions. Moreover, we are interested in extending our method to other downstream applications, such as event detection and activity recommendation.

**Acknowledgements.** This work is partially supported by National Natural Science Foundation of China (No. U163620068) and National Key Research and Development Program of China.

## References

1. Batmanghelich, K., Saeedi, A., Narasimhan, K., Gershman, S.: Nonparametric spherical topic modeling with word embeddings. arXiv preprint [arXiv:1604.00126](https://arxiv.org/abs/1604.00126) (2016)
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. *J. Mach. Learn. Res.* **3**(Jan), 993–1022 (2003)

3. Chen, W., Wang, J., Jiang, Z., Zhang, Y., Li, X.: Hierarchical mixed neural network for joint representation learning of social-attribute network. In: Kim, J., Shim, K., Cao, L., Lee, J.-G., Lin, X., Moon, Y.-S. (eds.) PAKDD 2017. LNCS (LNAI), vol. 10234, pp. 238–250. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-57454-7\\_19](https://doi.org/10.1007/978-3-319-57454-7_19)
4. Das, R., Zaheer, M., Dyer, C.: Gaussian LDA for topic models with word embeddings. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), vol. 1, pp. 795–804 (2015)
5. Duan, J., Ai, Y., et al.: LDA topic model for microblog recommendation. In: 2015 International Conference on Asian Language Processing (IALP), pp. 185–188. IEEE (2015)
6. Gopal, S., Yang, Y.: Von Mises-Fisher clustering models. In: International Conference on Machine Learning, pp. 154–162 (2014)
7. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864. ACM (2016)
8. Guo, J., Gong, Z.: A nonparametric model for event discovery in the geospatial-temporal space. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, pp. 499–508. ACM (2016)
9. Jabeen, I., Faiz, R., Mehmood, A., Yousaf, N.: Cohesion and semantic understanding. *Acad. Res. Int.* **4**(6), 139 (2013)
10. Kawamae, N.: Trend analysis model: trend consists of temporal words, topics, and timestamps. In: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, pp. 317–326. ACM (2011)
11. Krumm, J., Horvitz, E.: Eyewitness: identifying local events via space-time signals in Twitter feeds. In: Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems, p. 20. ACM (2015)
12. Liu, H., Ge, Y., Zheng, Q., Lin, R., Li, H.: Detecting global and local topics via mining Twitter data. *Neurocomputing* **273**, 120–132 (2018)
13. Liu, Y., Ester, M., Hu, B., Cheung, D.W.: Spatio-temporal topic models for check-in data. In: 2015 IEEE International Conference on Data Mining (ICDM), pp. 889–894. IEEE (2015)
14. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint [arXiv:1301.3781](https://arxiv.org/abs/1301.3781) (2013)
15. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710. ACM (2014)
16. Sizov, S.: Geofolk: latent spatial semantics in web 2.0 social media. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining, pp. 281–290. ACM (2010)
17. Stevens, K., Kegelmeyer, P., Andrzejewski, D., Buttler, D.: Exploring topic coherence over many models and many topics. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 952–961. Association for Computational Linguistics (2012)
18. Stieglitz, S., Mirbabaie, M., Ross, B., Neuberger, C.: Social media analytics-challenges in topic discovery, data collection, and data preparation. *Int. J. Inf. Manage.* **39**, 156–168 (2018)
19. Wang, J., Li, L., Tan, F., Zhu, Y., Feng, W.: Detecting hotspot information using multi-attribute based topic model. *PloS one* **10**(10), e0140539 (2015)

20. Wang, W., Yin, H., Chen, L., Sun, Y., Sadiq, S., Zhou, X.: Geo-sage: a geographical sparse additive generative model for spatial item recommendation. In: Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1255–1264. ACM (2015)
21. Wold, H.M., Vikre, L., Gulla, J.A., Özgöbek, Ö., Su, X.: Twitter topic modeling for breaking news detection. In: WEBIST, vol. 2. pp. 211–218 (2016)
22. Yan, X., Guo, J., Lan, Y., Cheng, X.: A bitern topic model for short texts. In: Proceedings of the 22nd International Conference on World Wide Web, pp. 1445–1456. ACM (2013)
23. Zhang, C., et al.: Triovecevent: embedding-based online local event detection in geo-tagged tweet streams. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 595–604. ACM (2017)
24. Zhang, K.: Uncovering urban dynamics via cross-modal representation learning. Ph.D. thesis (2017)
25. Zhao, W.X., et al.: Comparing Twitter and traditional media using topic models. In: Clough, P., et al. (eds.) ECIR 2011. LNCS, vol. 6611, pp. 338–349. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-20161-5\\_34](https://doi.org/10.1007/978-3-642-20161-5_34)



# Real-Time Route Planning and Online Order Dispatch for Bus-Booking Platforms

Hao Zhou, Yucen Gao, Xiaofeng Gao<sup>(✉)</sup>, and Guihai Chen

Shanghai Key Laboratory of Scalable Computing and Systems,  
Department of Computer Science and Engineering,  
Shanghai Jiao Tong University, Shanghai, China  
{h-zhou, guo\_ke}@sjtu.edu.cn, {gao-xf, g-chen}cs.sjtu.edu.cn

**Abstract.** To cater to the high travel demands in Beijing Capital International Airport during 23:00–2:00, Beijing Traffic Management Bureau (BTMB) intends to develop a new service named bus-booking platforms. Compared to traditional airport shuttle buses, bus-booking platforms can conduct flexible route planning and online order dispatch while provide much lower price than the common car-hailing platform, e.g., Didi Chuxing. We conduct the real-time route planning by solving the standard Capacitated Vehicles Routing Problem based on the order prediction. In addition, we focus on the design of the online order dispatch algorithm for bus-booking platforms, which is novel and extremely different from the traditional taxi order dispatch in car-hailing platforms. When an order is dispatched, multiple influence factors will be considered simultaneously, such as the bus capacity, the balanced distribution of the accepted orders and the travel time of passengers, all of which aim to provide a better user experience. Moreover, we prove that our online algorithms can achieve the tight competitive ratio in this paper, where the competitive ratio is the ratio between the solution of an online algorithm and the offline optimal solution.

**Keywords:** Order dispatch · Online algorithm · Intelligent transportation system

## 1 Introduction

The rapid deployment of transportation in recent years speeds up the travel between different cities and increases the travel frequency, which brings more traffic in return. To cater to the increasing travel demands and relieve the traffic pressure, some novel transportation concepts have been proposed and put

---

This work was supported by the National Key R&D Program of China [2018YFB1004703]; the National Natural Science Foundation of China [61872238, 61672353]; the Shanghai Science and Technology Fund [17510740200]; the Huawei Innovation Research Program [HO2018085286]; and the State Key Laboratory of Air Traffic Management System and Technology [SKLATM20180X].

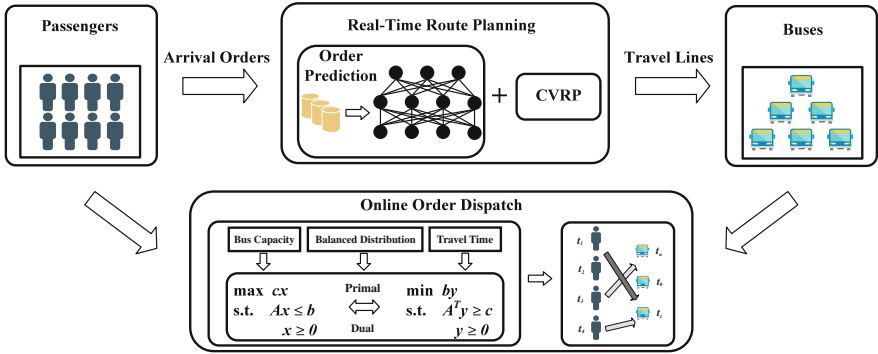


Fig. 1. The framework of bus booking platforms

into the wide application, e.g., online car-hailing. However, some locations with high travel demands are not fully satisfied yet. According to the official statistics from Beijing Traffic Management Bureau (BTMB), the number of the passengers arrived at Beijing Capital International Airport is about 18,000 during midnight period (23:00–02:00). However, the local public transport capacity can only provide pick-up service for around 8,200 passengers, including 7,000 passengers by taxi and online car-hailing, and 1,200 passengers by airport shuttle buses (less than 7% of the total arrived passengers), leaving a large number of passengers waiting at the airport for a long duration. In order to provide a higher-quality service for passengers, BTMB intends to develop a new service named bus-booking platforms. Unlike car-hailing platforms, bus-booking platforms can dispatch several orders to one bus with much lower price. Compared to traditional airport shuttle buses, bus-booking platforms can conduct the real-time route planning and the flexible order dispatch, which contribute to a better user experience.

As is shown in Fig. 1, the general framework of bus-booking platforms consists of two component: Real-Time Route Planning and Online Order Dispatch. When a bus is available in the source station, bus-booking platforms compute the travel line for this bus by solving the standard Capacitated Vehicle Routing Problem (CVRP). Once a new order arrives, bus-booking platforms will reply to the passenger whether the order is accepted and which bus the order is dispatched to if it is accepted. Although CVRP has been studied for many years, the previous works mainly concentrated on the offline cases where all the passengers and their destinations have been determined, which cannot directly apply to the real-time route planning [2,3,9]. In addition, the order dispatch problem for bus-booking platforms is novel and extremely different from the traditional taxi or car-hailing order dispatch problem, where the latter emphasizes on the matching of one order and one bus.

In this paper, we provide a complete overview for the bus-booking platform, from the framework design to the algorithm analysis. In order to overcome the problem where the common CVRP algorithms need to be given the destinations of passengers, we add the step of order prediction based on deep neural networks



(DNN). According to the arrived orders and historical data, bus-booking platforms will predict the basic attributes of new orders in next timeslot, containing the destination, the number of orders, etc. Besides the difference in the match pattern, most strategies for the order dispatch in car-hailing platforms are offline and heuristic, which cannot provide performance guarantee, e.g., the greedy mechanism in the early taxi-hailing platform [5], the offline Kuhn-Munkres (KM) algorithm [7] and the hill-climbing method deployed in Didi Chuxing [12, 13]. Different from them, we propose two effective online order dispatch algorithms for bus-booking platforms. It has been proved that our online algorithms achieves the tight competitive ratio, where the competitive ratio is the ratio between the solution of an online algorithm and the offline optimal solution. When conducting the order dispatch, bus-booking platforms will consider several influence factors simultaneously, e.g., the bus capacity, the balanced distribution of the accepted orders, etc. In order to improve the user experience, the travel time of passengers is also involved in the bus-booking platforms. Notice that it is necessary for passengers to wait some time for the reply in [12, 13] since the platform has to make decisions after aggregating all the arrived orders in one timeslot. However, our online algorithms can reply to the passengers right now, which does not depend on next arrived orders in the same timeslot.

## 2 Real-Time Route Planning

- **Order Prediction.** We divide the whole time period to many timeslot  $\langle t_1, t_2, \dots, t_n \rangle$ . Suppose that  $d$  buses come to the source station in timeslot  $t_k$ . We predict the new orders which arrives in timeslot  $t_k$  and compute the route line of  $d$  buses based on them. In the early stage of bus-booking platforms, the station set is given and passengers select a certain station as their destination. Let  $\mathcal{V}$  be the set of all the stations. Therefore, for each station  $v \in \mathcal{V}$ , we only need to predict the number of passengers who destined to this station in timeslot  $t_k$  in order to transform our problem to CVRP, which is denoted by  $\hat{x}_{t_k, v}$ . For convenience, we use the orders in the last  $h$  timeslots, i.e.,

$$\hat{x}_{t_k, v} = F_v(\hat{x}_{t_{k-h}, v}, \dots, \hat{x}_{t_{k-1}, v}), v \in \mathcal{V}$$

Here a deep neural network is utilized to predict the function  $F_v$ .

- **CVRP.** Capacitated Vehicle Routing Problem (CVRP) is to compute the travel lines for the vehicles with capacity so as to satisfy the requests of all the passengers and minimize the travel cost of all the vehicles. When the number of predicted orders is small, CVRP can be solved to compute a Capacitated Minimum Spanning Tree (CMST) because all the passengers depart from the same source station. Therefore, any  $\rho$ -approximation algorithm for CMST is exactly an  $2\rho$ -approximation algorithm for CVRP, where we transform each subtree in CMST to a tour by using the similar method in Travelling Salesman Problem [8]. If the number of orders is large, we have to reject some orders due to the limited number of buses. In this case, it is called Selective Vehicle Routing Problem and the genetic algorithms can be applied [2].

### 3 Online Order Dispatch

#### 3.1 Model and Problem Formulation

Suppose that the orders arrive one by one over time in the source station, e.g., Beijing Capital International Airport. Each order is a quadruple  $(i, d_i, c_i, t_i)$ , where  $i$  is the order number,  $d_i$  is the destination station,  $c_i$  is the number of passengers in this order and  $t_i$  is the arriving time. When an order arrives, there may be some available buses which are waiting to accept the orders again. Once a bus finishes its tour, it will return to the source station and wait the new orders. After an order arrives, we have to decide whether to accept this order and how to allocate it if it is accepted. Let  $x_{ij}$  denote whether the  $i$ -th order is dispatched to the  $j$ -th bus. Therefore, we have

$$\sum_{j \in B_i: d_i \in S_j} x_{ij} \leq 1, \forall i \in \mathcal{I} \quad (1)$$

where  $B_i$  is the set of the available buses in time  $t_i$ ,  $S_j$  is the set of the stations that the  $j$ -th bus passes and  $\mathcal{I}$  is the set of the orders. The passengers in the same order would like to take the same bus to the station. Let  $K_j$  be the number of seats in the  $j$ -th bus. Hence, the number of passengers allocated to the same bus should not exceed the capacity of this bus, i.e.,

$$\sum_{i \in I_j: d_i \in S_j} x_{ij} c_i \leq K_j, \forall j \in \mathcal{B} \quad (2)$$

where  $I_j$  denotes the set of the orders which arrive when the  $j$ -th bus are waiting in the source station and  $\mathcal{B}$  is the set of all the available buses throughout the time period. Due to the limited number of buses, we cannot provide the pick-up service for all the passengers and have to reject some orders. Let  $\mathcal{V}$  be the set of all the stations. In order to achieve the fairness, for each station  $v \in \mathcal{V}$ , there should be some orders which takes  $v$  as the destination to be accepted. In other words, the accepted orders should be balanced in all the stations instead of being centralized in a certain one. Therefore, we achieve this objective by using a bound  $R_v$  to limit the accepted orders or passengers destined to station  $v$ . The bound  $R_v$  can be determined based on the historical orders, the hot degree of each station, the total number of passengers which can be accepted by all the buses throughout the serving period, etc. Thus, we have

$$\sum_{i \in \mathcal{I}: d_i = v} \sum_{j \in B_i: v \in S_j} x_{ij} c_i \leq R_v, \forall v \in \mathcal{V} \quad (3)$$

Generally, the passengers would like to arrive their destinations as fast as possible and the user experience is influenced by the travel time, which should be considered in the allocation of orders. When an order arrives, there are several buses that can take the passenger to the destination station with different travel time. Let  $t_{ij}$  be the sum of the travel time of passengers in the  $i$ -th order by the

$j$ -th bus. In order to decrease the average travel time of all the passengers and improve the user experience, we bound the sum of the travel time of passengers for each bus and for each station, i.e.,

$$\sum_{i \in \mathcal{I}: d_i \in S_j} x_{ij} t_{ij} \leq T_j, \forall j \in \mathcal{B} \tag{4}$$

$$\sum_{i \in \mathcal{I}: d_i = v} \sum_{j \in \mathcal{B}: v \in S_j} x_{ij} t_{ij} \leq T_v, \forall v \in \mathcal{V} \tag{5}$$

The bound  $T_j$  can be obtained according to the capacity of the bus  $K_j$  and the travel time of its tour path, while the bound  $T_v$  can be determined based on  $R_v$  and the average travel time from the source station to the destination station  $v$ . Each order is usually assigned with a priority  $p_i$  to represent its importance. For example, the order that is generated by a VIP user has a higher priority and should be accepted with higher probability. In addition, the rejected user is allowed to send the order request again and waits the reply. Therefore, the higher priority should be assigned to the order with longer waiting time. Our objective is to maximize the sum of the priority of all the accepted orders. Notice that if  $p_i = 1$  for  $\forall i \in \mathcal{I}$ , the objective will be transformed to the maximization of the number of the accepted orders. Hence, we formulate our problem as a linear integer program as follows,

$$\begin{aligned} \max \quad & \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{B}: d_i \in S_j} x_{ij} p_i, \tag{6} \\ \text{s.t.} \quad & \text{Constraint (1), (2), (3), (4), (5)} \\ & x_{ij} \in \{0, 1\}, \forall i \in \mathcal{I}, j \in \mathcal{B} \end{aligned}$$

### 3.2 Online Algorithm Design

**Online Fractional Solution.** We first propose an online algorithm which produces a fractional solution and an integer solution can be obtained by using the randomized rounding technology. Before the further discussion, we present the dual of Program (6). We relax the integer constraint by replacing  $x_{ij} \in \{0, 1\}$  with  $x_{ij} \geq 0$  and take the dual variables for Constraints (1)–(5). Therefore, the dual objective is to minimize

$$\sum_{i \in \mathcal{I}} y_i + \sum_{j \in \mathcal{B}} (K_j z_j + T_j u_j) + \sum_{v \in \mathcal{V}} (R_v r_v + T_v q_v) \tag{7}$$

which subjects to

$$y_i + c_i(z_j + r_v) + t_{ij}(u_j + q_v) \geq p_i, \forall i \in \mathcal{I}, v = d_i, j \in B_i : d_i \in S_j \tag{8}$$

The Fractional Primal-Dual (FPD) algorithm is presented in Algorithm 1. All the dual variables are initialized as 0 in the beginning (lines 1–3). In the whole running process, Algorithm FPD ensures that the modified dual constraints

$$y_i + c_i(z_j + r_v) + t_{ij}(u_j + q_v) \geq p_i(1 - \exp(-\alpha)), \forall i \in \mathcal{I}, v = d_i, j \in B_i : d_i \in S_j$$

are always satisfied instead of Constraint (8), where  $\alpha$  is an argument which is used to help the proof of Theorem 1 and is determined by the competitive ratio  $\sigma$ . Once an order arrives, Algorithm FPD finds a modified dual constraint that is violated (line 7) and updates the dual variables based on  $x_{ij}$  until this constraint is satisfied (lines 8–15). All the dual variables are some functions of the primal variables  $x_{ij}$  and keep undiminished during the updates. Therefore, the modified dual constraint always holds once it has been satisfied. Notice that the coefficient in the update formula (lines 11–15) is not increasing, e.g.,  $p_{i^*}/c_{i^*}$ ,  $p_{i^*}/c_{i^*}$ ,  $p_{i^-}/t_{i^-j}$ ,  $p_{i\sim}/t_{i\sim j\sim}$ . Hence, the value of all the dual variables will not jump up during the updates, which will help our analysis for the competitive ratio. The argument  $\beta$  is similar to  $\alpha$  in Algorithm FPD (lines 11–15), which will be determined by the competitive ratio. The loop will terminate when no modified dual constraint is violated. Therefore, Algorithm FPD produces a feasible fractional solution for the dual program with the modified dual constraints. In addition, Algorithm FPD also provides a fractional solution for the primal program, which is taken as the output (line 18).

---

**Algorithm 1.** Fractional Primal-Dual (FPD) Algorithm

---

**Input:** The bus set  $\mathcal{B}$  and the station set  $\mathcal{V}$ .

**Output:** The fractional solution  $\{x_{ij}\}_{i \in \mathcal{I}, j \in \mathcal{B}}$ .

```

1: for  $i \in \mathcal{I}, j \in \mathcal{B}, v \in \mathcal{V}$  do
2:    $x_{ij} \leftarrow 0, y_i \leftarrow 0, z_j \leftarrow 0, u_j \leftarrow 0, r_v \leftarrow 0, q_v \leftarrow 0.$ 
3: end for
4:  $I_a \leftarrow \emptyset.$  /* the set of orders which have arrived */
5: while order  $(i, d_i, c_i, t_i)$  arrives do
6:    $I_a \leftarrow I_a \cup \{i\}.$ 
7:   while  $\exists j, y_i + c_i(z_j + r_v) + t_{ij}(u_j + q_v) < p_i(1 - \exp(-\alpha))$  do
8:     increase  $x_{ij}$  continuously.
9:      $y_i \leftarrow p_i \exp \left[ \alpha \left( \sum_{j \in \mathcal{B}: d_i \in S_j} x_{ij} - 1 \right) \right] - p_i \exp(-\alpha).$ 
10:     $i_* \leftarrow \arg \min_{i \in I_a \cap I_j: d_i \in S_j} \frac{p_i}{c_i}, \quad i_{\sim} \leftarrow \arg \min_{i \in I_a: d_i = v} \frac{p_i}{c_i}.$ 
11:     $z_j \leftarrow \max \left( \frac{p_{i_*}}{c_{i_*}} \left[ \exp \left( \beta \frac{\sum_{i \in I_j: d_i \in S_j} x_{ij} c_i}{2K_j} \right) - 1 \right], z_j \right).$ 
12:     $r_v \leftarrow \max \left( \frac{p_{i_*}}{c_{i_*}} \left[ \exp \left( \beta \frac{\sum_{i \in \mathcal{I}: d_i = v} \sum_{j \in \mathcal{B}: d_i \in S_j} x_{ij} c_i}{2R_v} \right) - 1 \right], r_v \right).$ 
13:     $i^- \leftarrow \arg \min_{i \in I_a \cap I_j: d_i \in S_j} \frac{p_i}{t_{ij}}, \quad (i\sim, j\sim) \leftarrow \arg \min_{i \in I_a, j \in \mathcal{B}: d_i = v \in S_j} \frac{p_i}{t_{ij}}.$ 
14:     $u_j \leftarrow \max \left( \frac{p_{i^-}}{t_{i^-j}} \left[ \exp \left( \beta \frac{\sum_{i \in I_j: d_i \in S_j} x_{ij} t_{ij}}{2T_j} \right) - 1 \right], u_j \right).$ 
15:     $q_v \leftarrow \max \left( \frac{p_{i\sim}}{t_{i\sim j\sim}} \left[ \exp \left( \beta \frac{\sum_{i \in \mathcal{I}: d_i = v} \sum_{j \in \mathcal{B}: d_i \in S_j} x_{ij} t_{ij}}{2T_v} \right) - 1 \right], q_v \right).$ 
16:   end while
17: end while
18: return  $\{x_{ij}\}_{i \in \mathcal{I}, j \in \mathcal{B}}.$ 

```

---

**Theorem 1.** *For any  $0 < \sigma < 1$ , Algorithm FPD produces an online fractional solution for the primal program, which achieves an  $\sigma$ -competitive ratio by setting  $\alpha = -\ln \sigma$  and  $\beta = \frac{1}{8} \ln^2 \sigma$ , but violates Constraint (2) for the  $j$ -th bus by at most  $O(\log \Gamma_j / \log^2 \sigma)$ , Constraint (3) for station  $v$  by at most  $O(\log \Lambda_v / \log^2 \sigma)$ , Constraint (4) for the  $j$ -th bus by at most  $O(\log \Upsilon_j / \log^2 \sigma)$  and Constraint (5) for station  $v$  by at most  $O(\log \Psi_v / \log^2 \sigma)$ , where*

$$\begin{aligned} \Gamma_j &= \max_{i \in I_j: d_i \in S_j} \left\{ \frac{p_i}{c_i} \right\} \times \max_{i \in I_j: d_i \in S_j} \left\{ \frac{c_i}{p_i} \right\}, \quad \Lambda_v = \max_{i \in \mathcal{I}: d_i = v} \left\{ \frac{p_i}{c_i} \right\} \times \max_{i \in \mathcal{I}: d_i = v} \left\{ \frac{c_i}{p_i} \right\} \\ \Upsilon_j &= \max_{i \in I_j: d_i \in S_j} \left\{ \frac{p_i}{t_{ij}} \right\} \times \max_{i \in I_j: d_i \in S_j} \left\{ \frac{t_{ij}}{p_i} \right\} \\ \Psi_v &= \max_{i \in \mathcal{I}, j \in B_i: d_i = v \in S_j} \left\{ \frac{p_i}{t_{ij}} \right\} \times \max_{i \in \mathcal{I}, j \in B_i: d_i = v \in S_j} \left\{ \frac{t_{ij}}{p_i} \right\} \end{aligned}$$

*Proof.* See Appendix A for the detailed proof.

Fix  $\sigma$  to a constant, e.g.,  $1/2$ , and let  $\Sigma = \max_{j,v} \{\Gamma_j, \Lambda_v, \Upsilon_j, \Psi_v\}$ . After Dividing  $K_j, R_v, T_j, T_v$  by  $\log \Gamma_j, \log \Lambda_v, \log \Upsilon_j, \log \Psi_v$  for Constraints (2)–(5) in the primal program and running Algorithm FPD again, we can obtain a feasible fractional solution for the primal program which achieves an  $O(\log \Sigma)$ -competitive ratio and does not violate any constraint.

**Improved Primal-Dual Algorithm.** Although an online integer solution can be obtained by using the randomized rounding technology for the fractional solution computed by Algorithm FPD, it only achieves an  $O(\log \Sigma \log N)$ -competitive ratio for the primal program, where  $N = |\mathcal{I}|$  is the number of orders. Next we propose an improved primal-dual algorithm, which always produces an integer solution.

As is shown in Algorithm 2, all the dual variables are initialized as 0 in the beginning (lines 1–3). Once a new order arrives, Algorithm IPD determines whether to accept this order based on the dual constraints. If the dual constraints for this order have been satisfied, Algorithm IPD directly rejects it and does not update any dual variables (lines 5–7). Otherwise, Algorithm IPD finds the  $j'$ -th bus that satisfies  $j' = \arg \min_{j \in B_i: d_i \in S_j} c_i z_j + t_{ij}(u_j + q_v)$  (line 5) and sets  $y_i \leftarrow 1 - c_i(z_{j'} + r_v) - t_{ij^*}(v_{j'} + q_v)$  (line 10), where the latter ensures that all the dual constraints for this order are satisfied again. Meanwhile, Algorithm IPD dispatches this new order to the  $j'$ -th bus (line 9) and updates all the dual variables based on the update formulas (lines 11–14). Similar to Algorithm FPD, the argument  $\epsilon$  in the update formulas is a constant that is determined by the competitive ratio.

**Theorem 2.** *For any constant  $\epsilon > 0$ , Algorithm IPD produces an online integer solution for the primal program, which achieves a  $(1 - \epsilon)$ -competitive ratio, but violates Constraint (2) for the  $j$ -th bus by at most  $O(\log \Gamma_j + \log 1/\epsilon)$ , Constraint (3) for station  $v$  by at most  $O(\log \Lambda_v + \log 1/\epsilon)$ , Constraint (4) for the  $j$ -th bus by at most  $O(\log \Upsilon_j + \log 1/\epsilon)$  and Constraint (5) for station  $v$  by at most  $O(\log \Psi_v + \log 1/\epsilon)$ , where*

---

**Algorithm 2.** Improved Primal-Dual (IPD) Algorithm

---

**Input:** The bus set  $\mathcal{B}$  and the station set  $\mathcal{V}$ .

**Output:** The integer solution  $\{x_{ij}\}_{i \in \mathcal{I}, j \in \mathcal{B}}$ .

```

1: for  $i \in \mathcal{I}, j \in \mathcal{B}, v \in \mathcal{V}$  do
2:    $x_{ij} \leftarrow 0, y_i \leftarrow 0, z_j \leftarrow 0, u_j \leftarrow 0, r_v \leftarrow 0, q_v \leftarrow 0.$ 
3: end for
4: while order  $(i, d_i, c_i, t_i)$  arrives do
5:    $j' \leftarrow \arg \min_{j \in \mathcal{B}: d_i \in S_j} c_i z_j + t_{ij}(u_j + q_v).$ 
6:   if  $c_i(z_{j'} + r_v) + t_{ij'}(u_{j'} + q_v) \geq p_i$  then
7:     reject order  $(i, d_i, c_i, t_i).$ 
8:   else
9:      $x_{ij'} \leftarrow 1.$ 
10:     $y_i \leftarrow p_i - c_i(z_{j'} + r_v) - t_{ij'}(u_{j'} + q_v).$ 
11:     $z_{j'} \leftarrow z_{j'} \left(1 + \frac{c_i}{K_{j'}}\right) + \frac{p_i \epsilon}{4K_{j'}}.$ 
12:     $r_v \leftarrow r_v \left(1 + \frac{c_i}{R_v}\right) + \frac{p_i \epsilon}{4R_v}.$ 
13:     $u_{j'} \leftarrow u_{j'} \left(1 + \frac{t_{ij'}}{T_{j'}}\right) + \frac{p_i \epsilon}{4T_{j'}}.$ 
14:     $q_v \leftarrow q_v \left(1 + \frac{t_{ij'}}{T_v}\right) + \frac{p_i \epsilon}{4T_v}.$ 
15:   end if
16: end while
17: return  $\{x_{ij}\}_{i \in \mathcal{I}, j \in \mathcal{B}}.$ 

```

---

$$\Gamma_j = \max_{i \in I_j: d_i \in S_j} \left\{ \frac{p_i}{c_i} \right\} \times \max_{i \in I_j: d_i \in S_j} \left\{ \frac{c_i}{p_i} \right\}, \quad A_v = \max_{i \in \mathcal{I}: d_i = v} \left\{ \frac{p_i}{c_i} \right\} \times \max_{i \in \mathcal{I}: d_i = v} \left\{ \frac{c_i}{p_i} \right\}$$

$$\Upsilon_j = \max_{i \in I_j: d_i \in S_j} \left\{ \frac{p_i}{t_{ij}} \right\} \times \max_{i \in I_j: d_i \in S_j} \left\{ \frac{t_{ij}}{p_i} \right\}$$

$$\Psi_v = \max_{i \in \mathcal{I}, j \in \mathcal{B}: d_i = v \in S_j} \left\{ \frac{p_i}{t_{ij}} \right\} \times \max_{i \in \mathcal{I}, j \in \mathcal{B}: d_i = v \in S_j} \left\{ \frac{t_{ij}}{p_i} \right\}$$

*Proof.* See Appendix B for the detailed proof.

### 3.3 The Lower Bound

Consider an extreme example where there are only one bus  $j$  and only one station  $v$ , and all the passengers would like to go to station  $v$ . Suppose that the number of passengers in each order and the travel time from the source station to station  $v$  are both one unit, i.e.,  $c_i = t_{ij} = 1, \forall i \in \mathcal{I}$ . In addition, we set  $K_j = R_v = T_j = T_v = K$  in this example. Suppose the orders arrive in batches, where each batch has  $K$  orders. For the  $i$ -th order, let  $p_i = 1/(M - m + 1)$  where  $M$  is a constant and this order locates in the  $m$ -th batch, i.e.,  $(m - 1)K < i \leq mK$ . Based on the above assumptions, this example can be formulated as follows,

$$\max \sum_{i \in \mathcal{I}} x_{ij} p_i, \tag{9}$$

$$\text{s.t. } \sum_{i \in \mathcal{I}} x_{ij} \leq K, \tag{9a}$$

$$x_{ij} \in \{0, 1\}, \forall i \in \mathcal{I} \tag{9b}$$

Notice that Constraint (2)–(5) in the primal program (6) are transformed to Constraint (9a) and Constraint (1) is equivalent to Constraint (9b) in this example. Even for this simple example, we claim that any online algorithm cannot provide a better performance guarantee.

**Theorem 3.** *For any  $0 < \rho \leq 1$ , a  $\rho$ -competitive online algorithm must violate Constraint (9a) by at least  $\rho \ln M$  in this example.*

*Proof.* Notice that the priority of an order is monotone increasing with the batch. When the  $m$ -th batch of orders arrives, the optimal solution is to accept all the orders in the  $m$ -th batch and reject all the former orders. Denote the optimal solution by  $\text{OPT}_{\mathcal{I}_m}$  where  $\mathcal{I}_m$  is the first  $m$  batch of orders. Denote by  $A_{\mathcal{I}_m}^\rho$  the solution of a  $\rho$ -competitive algorithm for  $\mathcal{I}_m$ . Therefore, we have

$$A_{\mathcal{I}_m}^\rho = \sum_{i \in \mathcal{I}_m} x_{ij} p_i \geq \rho \text{OPT}_{\mathcal{I}_m} = \frac{\rho K}{M - m + 1}$$

Accumulate the above equality from  $m = 1$  to  $M$  and let  $|\mathcal{I}| = MK$ . We have

$$\sum_{m=1}^M \sum_{i \in \mathcal{I}_m} x_{ij} p_i \geq \sum_{m=1}^M \frac{\rho K}{M - m + 1} = \rho K \ln M$$

Notice that

$$\begin{aligned} \sum_{m=1}^M \sum_{i \in \mathcal{I}_m} x_{ij} p_i &= \sum_{m=1}^M \sum_{l=1}^m \sum_{k=1}^K \frac{x_{ij}}{M - l + 1} = \sum_{l=1}^M \sum_{m=l}^M \sum_{k=1}^K \frac{x_{ij}}{M - l + 1} \\ &= \sum_{l=1}^M \sum_{k=1}^K x_{ij} = \sum_{i \in \mathcal{I}} x_{ij} \end{aligned}$$

Combining the above equalities, we finish the proof here.

According to Theorem 3, in order to obtain an online algorithm that does not violate any constraint, its competitive ratio  $\rho$  has to satisfy  $\rho \ln M \leq 1$ , i.e.,  $\rho \leq 1/\ln M$ . Notice that  $\Gamma_j = \Lambda_v = \Upsilon_j = \Psi_v = M$  in this example. Therefore, Algorithm IPD achieves the tight competitive ratio for the online order dispatch problem. In addition, the integer constraint (9b) is not used in the proof of Theorem 3, which means that the lower bound can also apply to the fractional solution of Program (9). Hence, Algorithm FPD also provides the tight performance guarantee in the competitive ratio for the fractional solution of the online order dispatch problem.

## 4 Experiment

### 4.1 Experimental Implementation

We conduct a comparative experiment to evaluate our online order dispatch algorithm. Top 30 hot stations are selected and the travel time between the stations are obtained according to their geographical distance. We randomly generate  $m$  buses and  $n$  orders during midnight period (23:00–02:00). Each bus arrives at Beijing Capital International Airport at random time point and departs from it after ten-minute waiting. The capacity of each bus is a parameter in our experiment. The travel line of each bus is computed by solving Capacitated Vehicle Routing Problem based on the arrived and predicted orders. The priority of each order is a random real number in  $(0, 1]$ . In order to provide a better user experience, four constraints have been considered in Program (6). The threshold value of the number of the accepted orders in one station is calculated by multiplying the total capacity of all the buses by the ratio between the number of orders destined to this station and the total number of all the orders. The threshold value of the travel time in one bus is calculated by multiplying the bus capacity by the average travel time of all the stations in this bus line. The threshold value of the travel time in one station is calculated based on the average travel time from the source station.

Based on the above statement, we make a detailed description of four compared algorithms involved in the experiment. *Random*: A random allocation algorithm. When an order arrives, a bus is randomly assigned to accept this order. We judge whether the above constraints can be satisfied after the bus accepts the order. If they are satisfied, the bus accepts this order and the state of the bus is updated, otherwise refuses. *Greedy*: A greedy algorithm based on the maximum-remaining-seat-first principle. When an order arrives, we find one bus with the most remaining seats that can take the passengers in this order to their destination station. We judge whether the above constraints can be satisfied after the bus accepts the order. If they are satisfied, the bus accepts the order and updates its state, otherwise refuses. *IPD*: The online order dispatch algorithm in this paper. *OPT*: The offline optimal solution by using Groubi 8.1.0 [1] to solve the linear integer program (6).

### 4.2 Experiment Result and Analysis

In the realistic situation, the total number of passengers included in the orders is often greater than the total capacity of all the buses due to the limited number of buses. Accordingly, we set different parameters and implement five experiments. The parameters in each experiment are shown in Table 1.

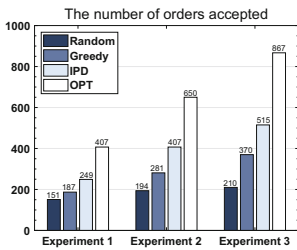
As is shown in Figs. 2, 3 and 4, both Random and Greedy do not perform well in the experiments. For the number of orders accepted, the number of passengers served and the priority of orders accepted, Random and Greedy only achieve less than 50% of the optimal solution compared with OPT. However, our online algorithm IPD can increase the results by more than 20% compared with Random



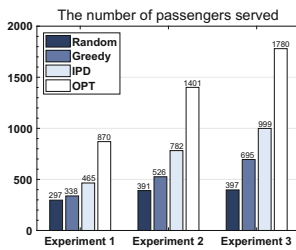
**Table 1.** Parameters in five experiments

Experiment	#Buses	Bus capacity	#Orders
1	30	30	1200
2	50	30	1200
3	50	40	1200
4	50	30	900
5	50	30	1500

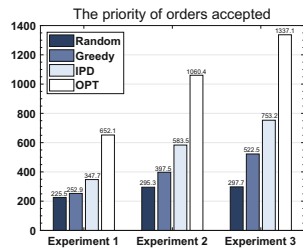
and Greedy. We only change the number of orders in Experiment 2, Experiment 4 and Experiment 5. According to Figs. 5, 6 and 7, we can find that the total number of orders during this period will not have a significant impact on the number of orders accepted, the number of passengers served and the priority of orders accepted if it is greater than the total bus capacity. Instead, the total bus capacity and the travel time constraint are the major influence factors when comparing the results in Figs. 2, 3, 4 and Figs. 5, 6, 7. Therefore, our online algorithm can provide the service for more passengers with almost the same total travel time compared with Random and Greedy. In other words, our algorithm can achieve lower average travel time. Although OPT performs much better in all the experiments, it requires too long running time and to be given all the orders in advance, which cannot apply to online situations.



**Fig. 2.** Orders



**Fig. 3.** Passengers



**Fig. 4.** Priority

## 5 Related Work

There are many works about the order dispatch problem for car-hailing platforms. Based on global positioning systems (GPS), the greedy mechanism was proposed for real-time taxi order dispatch where the order was matched with the nearest driver [5]. Although the above greedy strategy could significantly decrease the waiting time of passengers and improve the user experience, it did not consider the global profits of platforms and taxis. The queuing strategy based on the first-come-first-serve principle was also widely used in the early taxi

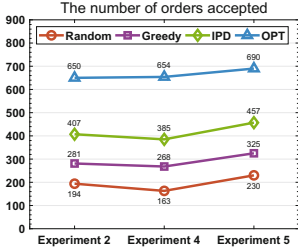


Fig. 5. Orders

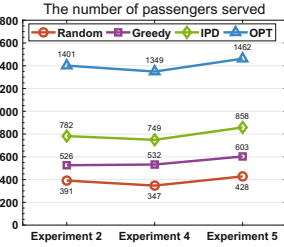


Fig. 6. Passengers

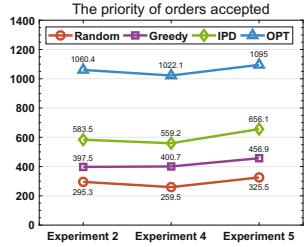


Fig. 7. Priority

platforms [4, 10, 14]. The offline Kuhn-Munkres (KM) algorithm [7] was deployed in the Didi Chuxing platform, which collected all the arrived orders and the available cars in one timeslot and conducted the maximum bipartite matching [12]. In order to increase the total acceptance rate of orders, another work attempted to dispatch one order to several drivers, where the first one to accept the requirement obtained this order [13]. However, both of them focused on heuristic or offline algorithms, which could not provide the global performance guarantee. In addition, there were also some works that used the demand prediction to help the order dispatch [6, 11], but all of them concentrated on taxi or car-hailing platforms.

## 6 Conclusion

In this paper, we presented a complete overview for a new service named bus-booking platforms. Real-time route planning in bus-booking platforms can be achieved by solving the standard CVRP based on the order prediction. In addition, we proposed two novel online order dispatch algorithms to satisfy the real-time requirements from customers. We proved that our online algorithms can provide solid theoretical guarantee and touch the tight competitive ratio, where the competitive ratio is the ratio between the solution of an online algorithm and the offline optimal solution.

## A Proof of Theorem 1

*Proof.* Consider a modified primal program as follows and let  $OPT, OPT_\alpha$  be the optimal objective value for the primal program (6) and the modified program (10), respectively. Obviously, we have  $OPT_\alpha \geq (1 - \exp(-\alpha))OPT$ .

$$\begin{aligned}
 & \max \sum_{i \in \mathcal{I}} \sum_{j \in B_i : d_i \in S_j} x_{ij} p_i (1 - \exp(-\alpha)), \\
 & \text{s.t. Constraint (1), (2), (3), (4), (5)} \\
 & \quad x_{ij} \geq 0, \forall i \in \mathcal{I}, j \in \mathcal{B}
 \end{aligned} \tag{10}$$

Notice that Algorithm FPD exactly produces a feasible fractional solution for the dual of Program (10), where denote by  $D_\alpha$  its objective value. Let  $P$  be the objective value computed by Algorithm FPD for the primal program (6).

Therefore, we have  $\frac{\partial P}{\partial x_{ij}} = p_i$  and

$$\frac{\partial D_\alpha}{\partial x_{ij}} = \frac{\partial y_i}{\partial x_{ij}} + K_j \frac{\partial z_j}{\partial x_{ij}} + T_j \frac{\partial u_j}{\partial x_{ij}} + R_v \frac{\partial r_v}{\partial x_{ij}} + T_v \frac{\partial q_v}{\partial x_{ij}} \quad (\text{Note: } v = d_i)$$

From line 9 in Algorithm FPD, we get

$$\frac{\partial y_i}{\partial x_{ij}} = \alpha p_i \exp \alpha \left( \sum_{j \in B_i: d_i \in S_j} x_{ij} - 1 \right)$$

Since  $x_{ij}$  increases continuously and the condition in line 7 in Algorithm FPD is satisfied,  $y_i \leq p_i(1 - \exp(-\alpha))$  is always guaranteed. Combining line 9, we have

$\frac{\partial y_i}{\partial x_{ij}} \leq \alpha p_i$ . Similarly, we can get

$$\frac{\partial z_j}{\partial x_{ij}} \leq \frac{\beta c_i p_{i^*}}{2K_j c_{i^*}} \exp \beta \left( \frac{\sum_{i \in I_j: d_i \in S_j} x_{ij} c_i}{2K_j} \right)$$

Due to  $c_i z_j \leq p_i(1 - \exp(-\alpha)) \leq p_i$ , we have

$$\frac{\partial z_j}{\partial x_{ij}} \leq \frac{\beta c_i}{2K_j} \left( \frac{p_i}{c_i} + \frac{p_{i^*}}{c_{i^*}} \right) \leq \frac{\beta c_i}{2K_j} 2 \frac{p_i}{c_i} = \frac{\beta p_i}{K_j}$$

By the same way, we can get  $\frac{\partial r_v}{\partial x_{ij}} \leq \frac{\beta p_i}{R_v}$ ,  $\frac{\partial u_j}{\partial x_{ij}} \leq \frac{\beta p_i}{T_j}$ ,  $\frac{\partial q_v}{\partial x_{ij}} \leq \frac{\beta p_i}{T_v}$ . Therefore,

we have  $\frac{\partial D_\alpha}{\partial x_{ij}} \leq (\alpha + 4\beta)p_i = (\alpha + 4\beta) \frac{\partial P}{\partial x_{ij}}$ . From the weak duality property,

we have  $\text{OPT}_\alpha \leq D_\alpha \leq (\alpha + 4\beta)P$ . Therefore, we further get

$$P \geq \frac{\text{OPT}_\alpha}{\alpha + 4\beta} \geq \frac{1 - \exp(-\alpha)}{\alpha + 4\beta} \text{OPT}$$

Let  $\sigma = (1 - \exp(-\alpha))/(\alpha + 4\beta)$  and  $\alpha = -\ln \sigma$ . We can get

$$\beta = \frac{1 - \sigma + \sigma \ln \sigma}{4\sigma}$$

Let  $g(\sigma) = 1 - \sigma + \sigma \ln \sigma - \frac{1}{2}\sigma \ln^2 \sigma$ . Since  $g'(\sigma) = -\frac{1}{2}\ln^2 \sigma \leq 0$  and  $\sigma \in (0, 1]$ , we have  $g(\sigma) \geq g(1) = 0$ . Therefore, we can get

$$\beta \geq \frac{\frac{1}{2}\sigma \ln^2 \sigma}{4\sigma} = \frac{1}{8} \ln^2 \sigma$$

Now consider all the constraints in the primal program (6). First Constraint (1) is always satisfied because  $y_i \leq p_i(1 - \exp(-\alpha))$  always holds. Let  $i^* = \arg \max_{i \in I_j : d_i \in S_j} \frac{p_i}{c_i}$ . Because  $z_j$  is increased continuously and Algorithm FPD will not increase  $z_j$  if the condition in line 7 is not satisfied, we have  $z_j \leq \frac{p_{i^*}}{c_{i^*}}$ , i.e.,

$$\frac{p_{i^*}}{c_{i^*}} \left[ \exp \left( \beta \frac{\sum_{i \in I_j : d_i \in S_j} x_{ij} c_i}{2K_j} \right) - 1 \right] \leq \frac{p_{i^*}}{c_{i^*}}$$

Rearranging the above inequality, we have

$$\frac{\sum_{i \in I_j : d_i \in S_j} x_{ij} c_i}{K_j} \leq \frac{2}{\beta} \log \left( \frac{p_{i^*}}{c_{i^*}} \frac{p_{i^*}}{c_{i^*}} + 1 \right) \leq \frac{16}{\ln^2 \sigma} \log \left( \frac{p_{i^*}}{c_{i^*}} \frac{p_{i^*}}{c_{i^*}} + 1 \right) = O \left( \frac{\log \Gamma_j}{\log^2 \sigma} \right)$$

By the same way, we finish the proof.

## B Proof of Theorem 2

*Proof.* Let  $x$  and  $(y, z, r, u, q)$  be the primal and dual solution computed by Algorithm IPD while  $P$  and  $D$  be the objective values, respectively. We finish the proof of this theorem from the following two properties:

- (1) The solution  $(y, z, r, u, q)$  is feasible for the dual program.
- (2) When order  $(i, d_i, c_i, t_i)$  is accepted, the objective of the dual program increases  $(1 + \epsilon)p_i$ .

For Property (1), if order  $(i, d_i, c_i, t_i)$  is rejected, we have  $c_i(z_j + r_v) + t_{ij}(u_j + q_v) \geq p_i$  for  $\forall j \in B_i : d_i \in S_j$ . Otherwise, we set  $y_i \leftarrow p_i - c_i(z_{j'} + r_v) - t_{ij'}(v_{j'} + q_v)$ , which guarantees that  $y_i + c_i(z_j + r_v) + t_{ij}(u_j + q_v) \geq p_i$  for  $\forall j : d_i \in S_j$  because  $j' \leftarrow \arg \min_{j \in B_i : d_i \in S_j} c_i z_j + t_{ij}(u_j + q_v)$ .

For Property (2), when order  $(i, d_i, c_i, t_i)$  is accepted, all the dual variables will be updated. Therefore, the objective of the dual program increases

$$\begin{aligned} \Delta D &= \Delta y_i + \Delta K_{j'} z_{j'} + \Delta R_v r_v + \Delta T_{j'} u_{j'} + \Delta T_v q_v \\ &= p_i - c_i(z_{j'} + r_v) - t_{ij'}(v_{j'} + q_v) + K_{j'} \left( \frac{z_{j'} c_i}{K_{j'}} + \frac{p_i \epsilon}{4K_{j'}} \right) \\ &\quad + R_v \left( \frac{r_v c_i}{R_v} + \frac{p_i \epsilon}{4R_v} \right) + T_{j'} \left( \frac{u_{j'} t_{ij'}}{T_{j'}} + \frac{p_i \epsilon}{4T_{j'}} \right) + T_v \left( \frac{q_v t_{ij'}}{T_v} + \frac{p_i \epsilon}{4T_v} \right) \\ &= (1 + \epsilon)p_i \end{aligned}$$

Based on Property (1) (2) and the weak duality, we get

$$P \geq \frac{D}{1 + \epsilon} \geq \frac{\text{OPT}}{1 + \epsilon} \geq (1 - \epsilon)\text{OPT}$$

For the constraints, we consider the dual variables  $(z, r, u, q)$ , respectively. Let  $z_j^t$  be the value of  $z_j$  after the  $t$ -th update. Let  $i_* = \arg \min_{i: d_i \in S_j} \frac{p_i}{c_i}$ . Therefore, we have

$$z_j^t = z_j^{t-1} \left( 1 + \frac{c_i}{K_j} \right) + \frac{p_i \epsilon}{4K_j} \geq z_j^{t-1} \left( 1 + \frac{c_i}{K_j} \right) + \frac{c_i \epsilon}{4K_j} \frac{p_{i_*}}{c_{i_*}}$$

Rearranging the above inequality, we can get

$$z_j^t + \frac{p_{i_*} \epsilon}{4c_{i_*}} \geq \left( z_j^{t-1} + \frac{p_{i_*} \epsilon}{4c_{i_*}} \right) \left( 1 + \frac{c_i}{K_j} \right)$$

Let  $\gamma_j$  satisfy

$$\left( 1 + \frac{c_i}{K_j} \right) \geq \gamma_j^{\frac{c_i}{K_j}}, \quad \forall i : d_i \in S_j \tag{11}$$

Therefore, we have

$$\begin{aligned} z_j^t + \frac{p_{i_*} \epsilon}{4c_{i_*}} &\geq \left( z_j^{t-1} + \frac{p_{i_*} \epsilon}{4c_{i_*}} \right) \gamma_j^{\frac{c_i}{K_j}} \\ &\geq \left( z_j^0 + \frac{p_{i_*} \epsilon}{4c_{i_*}} \right) \gamma_j^{\frac{\sum_{i: d_i \in S_j} x_{ij} c_i}{K_j}} \\ &= \frac{p_{i_*} \epsilon}{4c_{i_*}} \gamma_j^{\frac{\sum_{i \in I_j: d_i \in S_j} x_{ij} c_i}{K_j}} \end{aligned} \tag{12}$$

According to Algorithm IPD (lines 6–7), before the last update of  $\tilde{z}_j$ , we have

$$c_i(z_j^{t-1} + q_v^{t-1}) + t_{ij}(u_j^{t-1} + q_v^{t-1}) < p_i$$

Hence,  $z_j^{t-1} < \frac{p_i}{c_i}$ . Let  $i^* = \arg \max_{i \in B_j: d_i \in S_j} \frac{p_i}{c_i}$ . After the last update, we can get

$$\begin{aligned} z_j^t &< \frac{p_i}{c_i} \left( 1 + \frac{c_i}{K_j} \right) + \frac{p_i \epsilon}{4K_j} = \frac{p_i}{c_i} + \left( 1 + \frac{\epsilon}{4} \right) \frac{p_i}{K_j} \leq \frac{p_i}{c_i} + \left( 1 + \frac{\epsilon}{4} \right) \frac{p_i}{c_i} \\ &\leq \frac{p_{i^*}}{c_{i^*}} \left( 2 + \frac{\epsilon}{4} \right) \end{aligned}$$

Combining Inequality (12), we have

$$\frac{p_{i^*}}{c_{i^*}} \left( 2 + \frac{\epsilon}{4} \right) \geq z_j^t \geq \frac{p_{i_*} \epsilon}{4c_{i_*}} \left( \gamma_j^{\frac{\sum_{i \in I_j: d_i \in S_j} x_{ij} c_i}{K_j}} - 1 \right)$$

Therefore,

$$\frac{\sum_{i \in I_j: d_i \in S_j} x_{ij} c_i}{K_j} \leq \log_{\gamma_j} \left( \frac{p_{i^*} c_{i_*}}{c_{i^*} p_{i_*}} \left( \frac{8}{\epsilon} + 1 \right) + 1 \right) \leq \log_{\gamma_j} \frac{p_{i'} c_{i^*}}{c_{i'} p_{i^*}} \left( \frac{8}{\epsilon} + 2 \right)$$

Now consider  $\gamma_j$ . From Equality (11), we have

$$\ln \gamma_j \leq \min_{i \in I_j: d_i \in S_j} \frac{\ln(1 + \frac{c_i}{K_j})}{\frac{c_i}{K_j}}$$

Because  $0 \leq \frac{c_i}{K_j} \leq 1$  and  $f(x) = \frac{\ln(1+x)}{x}$  is a monotone decreasing function, Equality (11) is satisfied when  $\gamma_j = 2$ . Therefore,

$$\frac{\sum_{i \in I_j: d_i \in S_j} x_{ij} c_i}{K_j} \leq \left( \log_2 \frac{p_{i^*} c_{i^*}}{c_{i^*} p_{i^*}} \left( \frac{8}{\epsilon} + 2 \right) \right) = O \left( \log \Gamma_j + \log \frac{1}{\epsilon} \right)$$

By the same way, we finish this proof.

## References

1. Gurobi. <http://www.gurobi.com/>
2. Allahviranloo, M., Chow, J.Y., Recker, W.W.: Selective vehicle routing problems under uncertainty without recourse. *Transp. Res. Part E Logist. Transp. Rev.* **62**, 68–88 (2014)
3. Cáceres-Cruz, J., Arias, P., Guimarans, D., Riera, D., Juan, A.A.: Rich vehicle routing problem: survey. *ACM Comput. Surv.* **47**(2), 32:1–32:28 (2014)
4. Lee, D.-H., Wang, H., Cheu, R.L., Teo, S.H.: Taxi dispatch system based on current demands and real-time traffic conditions. *Transp. Res. Rec. J. Transp. Res. Board* **1882**, 193–200 (2004)
5. Liao, Z.: Real-time taxi dispatching using global positioning systems. *Commun. ACM (CACM)* **46**(5), 81–83 (2003)
6. Moreira-Matias, L., Gama, J., Ferreira, M., Mendes-Moreira, J., Damas, L.: Predicting taxi-passenger demand using streaming data. *IEEE Trans. Intell. Transp. Syst. (TITS)* **14**(3), 1393–1402 (2013)
7. Munkres, J.: Algorithms for the assignment and transportation problems. *J. Soc. Ind. Appl. Math.* **5**(1), 32–38 (1957)
8. Papadimitriou, C.H., Steiglitz, K.: *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Englewood Cliffs (1982)
9. Ralphs, T.K., Kopman, L., Pulleyblank, W.R., Trotter, L.E.: On the capacitated vehicle routing problem. *Math. Program.* **94**(2–3), 343–359 (2003)
10. Seow, K.T., Dang, N.H., Lee, D.: A collaborative multiagent taxi-dispatch system. *IEEE Trans. Autom. Sci. Eng. (TASAE)* **7**(3), 607–616 (2010)
11. Tong, Y., et al.: The simpler the better: a unified approach to predicting original taxi demands based on large-scale online platforms. In: *ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 1653–1662 (2017)
12. Xu, Z., et al.: Large-scale order dispatch in on-demand ride-hailing platforms: a learning and planning approach. In: *ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 905–913 (2018)
13. Zhang, L., et al.: A taxi order dispatch model based on combinatorial optimization. In: *ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 2151–2159 (2017)
14. Zhang, R., Pavone, M.: Control of robotic mobility-on-demand systems: a queueing-theoretical perspective. *Int. J. Robot. Res.* **35**(1–3), 186–203 (2016)



# STL: Online Detection of Taxi Trajectory Anomaly Based on Spatial-Temporal Laws

Bin Cheng<sup>1,2</sup>, Shiyou Qian<sup>1,2(✉)</sup>, Jian Cao<sup>1,2</sup>, Guangtao Xue<sup>1</sup>, Jiadi Yu<sup>1</sup>, Yanmin Zhu<sup>1</sup>, Minglu Li<sup>1</sup>, and Tao Zhang<sup>3,4</sup>

<sup>1</sup> Department of Computer Science and Engineering,  
Shanghai Jiao Tong University, Shanghai, China

{bincheng, qshiyou, cao-jian, gt\_xue, jiadiyu, yzhu, mlli}@sjtu.edu.cn

<sup>2</sup> Shanghai Institute for Advanced Communication and Data Science,  
Shanghai Jiao Tong University, Shanghai, China

<sup>3</sup> Department of Computer Engineering and Science,  
Shanghai University, Shanghai, China

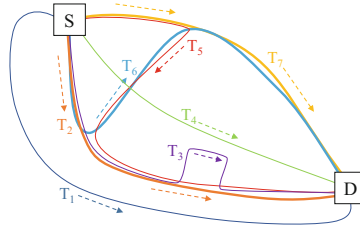
<sup>4</sup> Shanghai Institute for Advanced Communication and Data Science,  
Shanghai University, Shanghai, China  
taozhang@shu.edu.cn

**Abstract.** Aiming to promote the standardization of taxi services and protect the interests of passengers, many methods have been proposed to detect taxi trajectory anomaly based on collected large-scale taxi traces. However, most existing methods usually employ a counting-based policy to differentiate normal trajectories from anomalous ones, which may give rise to high false positives. In this paper, we propose an online detection method, named Spatial-Temporal Laws (STL). The basic idea of STL is that, given the displacement from the source point to the current point of a testing trajectory, if the current point is normal, either its driving distance or driving time should lie in a normal range. STL learns the two ranges from historical trajectories by defining two spatial-temporal models: one characterizing the relationship between displacement and driving distance, and another depicting the relationship between displacement and driving time. Consequently, STL is more precise compared with the counting-based methods, greatly reducing the number of false positives. Based on large-scale real-world taxi traces, STL is evaluated through a series of experiments which demonstrate its effectiveness and performance.

**Keywords:** Anomaly detection · Taxi trajectory · Spatial-temporal

## 1 Introduction

In recent years, many novel methods have been proposed to detect taxi trajectory anomaly [1–3, 9, 10, 12–14], highlighting their practical value in terms of saving passengers' time and money. Most existing methods utilize a counting policy to detect anomalous taxi trajectories [1–3, 12–14]. The rationale underpinning these methods is that trajectories that have been seldom traversed by taxis in



**Fig. 1.** Example of taxi trajectories between S and D.

the past are considered to be anomalous. Taking Fig. 1 as an example, where S and D represent the source and the destination point respectively, of the seven trajectories shown in the figure, counting-based methods identify  $T_2$ ,  $T_6$  and  $T_7$  as normal trajectories since they are traversed by most taxis. As normal trajectories appear a significantly greater number of times in the set of historical trajectories,  $T_1$ ,  $T_3$ ,  $T_4$  and  $T_5$  are labelled anomalous, even though  $T_4$  is shorter than the three common trajectories.

It is clear that counting-based detection methods have a major drawback in that they result in a high number of false positives. Furthermore, when detecting a testing trajectory starting from a source to a destination point, if there are fewer historical trajectories with the same source and destination point, counting-based methods may consider the trajectory as anomalous, which means that these methods need a large number of historical trajectories to work properly.

In essence, a trajectory that costs passengers more in both driving time and driving distance should be considered as anomalous. As shown in Fig. 1, the driving distance of both  $T_4$  and  $T_5$  is not greater than the three normal trajectories, and  $T_3$  may be the result of an experienced taxi driver choosing an unusual detour to avoid congestion. In reality,  $T_3$ ,  $T_4$  and  $T_5$  should not be considered to be anomalous since the cost (either driving distance or driving time) is the same as  $T_2$ ,  $T_6$  and  $T_7$ .

This paper proposes a novel taxi trajectory anomaly detection method, called STL, based on the essential spatial-temporal laws of taxi transportation. The basic idea of STL is that, for a displacement calculated from any point of a normal trajectory to its source point, the costs paid, measured by driving distance and driving time, should lie within normal ranges. STL learns two spatial-temporal models that define two normal ranges from historical taxi trajectories: the D-S model characterizing the relationship between displacement (for short D) and driving distance (S), and the D-T model characterizing the relationship between displacement and driving time (T). Given the two normal ranges, for any point of a testing trajectory, the driving distance and driving time incurred for the displacement are compared to the normal ranges. If both costs are beyond the normal range, it is more likely to be anomalous.

Compared with counting-based methods, STL has three main advantages: (1) it can efficiently detect anomalies in a stream of trajectories online, which means it can immediately respond to the updated taxi traces; (2) taking into



consideration the relationships between spatial and temporal features, STL can detect diverse but not malicious trajectories correctly, producing a low number of false positives; and (3) STL is more general, because the transportation characteristics of adjacent areas are similar so their historical trajectories can be gathered to learn the D-S and D-T models, conquering the limitation of existing methods that demand a large volume of historical trajectories to cover the road network.

We evaluate STL on large-scale real-world taxi traces collected from three different cities in one month. In the experiments, we evaluate the performance of STL on different kinds of taxi trajectories in comparison with three other anomalous taxi trajectory detection methods. The results show that STL achieves an excellent performance with a high true positive rate and a low false positive rate in different situations.

## 2 Problem Definition

On the two-dimensional plane (longitude and latitude), a taxi trajectory can be represented as a collection of points in chronological order. For instance, Fig. 2 shows the trajectory of a taxi in a day, where the red and green points represent the taxi's status in terms of vacant or occupied, respectively. Since the fraud behaviors of taxis always occur when taxis are occupied, in this paper, we only consider the trajectories of the taxis that are occupied.

**Definition 1** (*Point*). Given a record generated by a GPS device, let  $x, y$  be the longitude and latitude of the location and  $ts$  be the timestamp of the record. In this way, the spatial information and temporal information of a record can be represented by a point in the form of a triple  $(x, y, ts)$ .

**Definition 2** (*Trajectory*). A trajectory  $T$  consists of a collection of points in chronological order,  $T = \langle p_1, p_2, \dots, p_n \rangle$ . For any  $1 \leq i < j \leq n$ , the timestamp of  $p_i$  is smaller than the one of  $p_j$  and  $T_{i \rightarrow j}$  denotes a sub-trajectory  $\langle p_i, \dots, p_j \rangle$  of  $T$ .

**Definition 3** (*Displacement, Driving-Distance, Driving Time*). No matter whether a trajectory is completed or not, the source point of the trajectory, denoted as  $p_1$ , is known in advance. For a point  $p_i$  observed currently, the displacement, denoted by  $d_i$ , from  $p_i$  to the source point  $p_1$  is defined as

$$d_i = \text{dist}(p_i, p_1)$$

where  $\text{dist}(p_i, p_1)$  is a function that calculates the geographical distance between the two points.

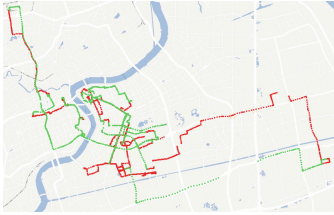
The driving distance of  $p_i$ , denoted as  $s_i$ , is the summation of the geographical distance of all the segments before  $p_i$ , which is computed as

$$s_i = \sum_{j=1}^{i-1} \text{dist}(p_j, p_{j+1})$$

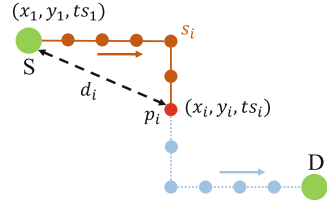
Also, we define the driving time, denoted by  $t_i$ , from point  $p_i$  to the source point  $p_1$  as

$$t_i = p_i.ts - p_1.ts$$

As shown in Fig. 3, the displacement of  $p_i$  is the length of the black dashed line and the driving-distance of  $p_i$  is the length of the brown line.



**Fig. 2.** One taxi trajectory in a day. (Color figure online)



**Fig. 3.** An illustrated example of  $d$  and  $s$ . (Color figure online)

**Definition 4 (Map function).** When detecting a testing trajectory, it is necessary to get all the historical trajectories with the same source and destination point as the trajectory. With a huge number of trajectories, the search cost is intolerable without any pre-processing. In order to retrieve historical trajectories more efficiently, we split the road network into equal-sized grid cells and transform a point to a cell by a map function which is defined as

$$g_i = \text{map}(p_i) : R^2 \rightarrow G$$

By mapping the coordinates of a point to a grid, a point trajectory (consisting of points) is transformed to a grid-cell trajectory (composed of grid cells). In this way, it is more efficient to identify the trajectories which pass a specific point by searching the trajectories that pass the corresponding grid cell with the help of an inverted index.

**Definition 5 (Problem Definition).** We define the problem addressed in this paper as follows: Given a set of historical trajectories  $S = \{T_1, T_2, \dots\}$  and a testing trajectory  $T = \{p_1, p_2, \dots\}$  with the source point  $p_1$  and the destination point  $p_d$ , as passengers who use apps like Uber<sup>1</sup> and Didi<sup>2</sup> to call taxis input their current location and destination, the problem of online trajectory anomaly detection is to find the set of outlier points  $P = \{p_i | p_i \in T\}$  in  $T$ .

### 3 Related Work

In general, existing anomaly detection methods are roughly classified into two categories: detection based on spatial features and detection based on temporal features.

<sup>1</sup> <https://www.uber.com/>.  
<sup>2</sup> <https://www.didiglobal.com/>.

### 3.1 Detection Based on Spatial Features

This type of detection makes use of spatial features to detect anomalies. More specifically, according to the timeliness, this class can be further divided into two sub-categories: offline detection and online detection.

**Offline Detection.** Lee et al. proposed a novel partition-and-detect framework for trajectory outlier detection which first partitions a trajectory into a set of line segments and then detects outlying segments by a hybrid of the distance-based and density-based approaches [4]. Ge et al. developed a taxi driving fraud detection system based on travel route evidence and driving distance evidence [3]. Zhang et al. proposed an algorithm called iBAT based on the idea that anomalous trajectory is easy to isolate by detecting the complete trajectories offline [11]. Liu et al. proposed an abnormal behavior detection method that is independent of map information and road networks by detecting inconsistencies between the velocity of the vehicle and the speed information of the GPS [6].

**Online Detection.** Based on iBAT [11], Chen et al. proposed the iBOAT detection method [2]. They considered a trajectory with lower support from historical trajectories as anomalous. It has a higher detection accuracy compared to iBAT and can be performed online. Zhou et al. proposed a detection method named OnATrade [12], consisting of two steps: route recommendation and online detection.

### 3.2 Detection Based on Temporal Features

The second class makes use of temporal features to detect trajectory. For instance, Li et al. proposed a method for detecting temporal outliers with an emphasis on historical similarity trends between data points [5]. Zhu et al. proposed a time-dependent popular route-based algorithm [14] and designed a time-dependent transfer graph from which the top-k most popular routes are obtained as references in different time periods. Based on [14], Zhu et al. presented an upgraded version of TPRO, named TPRRO [13], which is a real-time outlier detection method. They developed a time-dependent transfer index and a hot time-dependent transfer graph cache which speed up the online detection progress.

As discussed, most existing detection methods only separately consider the spatial or temporal features of trajectories, producing high false positives. Furthermore, the existing methods require a large number of historical trajectories to work properly, and do not have the ability to deal with cases where trajectories are sparse in some areas. In addition, there is scant research on online detection especially in relation to taxis. Therefore, further research into online taxi trajectory detection is urgently needed, which motivates our work.

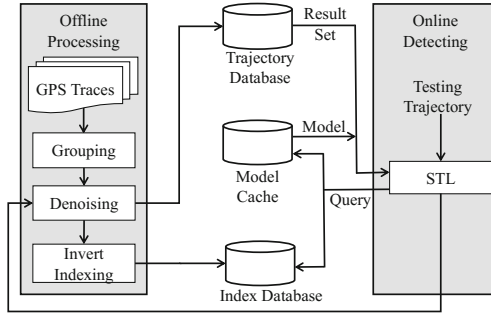
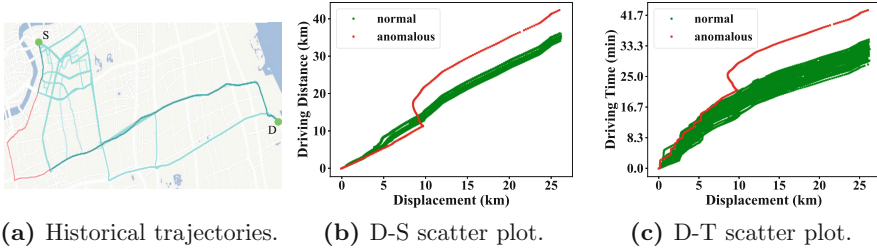


Fig. 4. Trajectory detection framework.

## 4 Detection Framework

This section provides the trajectory detection framework which is shown in Fig. 4. The whole detection process is divided into two stages: offline pre-processing and online detection. The role of each component in this framework is described as follows:

- **Taxi Traces:** Taxi traces refer to the collection of GPS records uploaded periodically to the server by all taxis.
- **Grouping:** Since the records of all taxis are mixed together in the traces, it is necessary to group them according to the taxi IDs and sort the GPS records by the timestamp in each group. After sorting the GPS points of each taxi, the parts of trajectory when the taxi is occupied are extracted.
- **Denoising:** In order to reduce the noise in the set of extracted trajectories, two adjacent points over 5 km away will be separated into two trajectories and the point which causes the speed between it and the last point to be larger than 120 km/h will be ignored. Furthermore, the map-matching technology [7] is applied to project the trajectory points into the road network.
- **Invert Indexing:** The number of trajectories extracted from the taxi traces is very large. To improve the performance for searching the trajectories that pass a certain grid-cell, the inverted index which records the trajectories passing the grid-cell and the index in which the grid appears is constructed.
- **Trajectory Database:** This stores all the point trajectories extracted from the traces and the corresponding grid-cell trajectories mapped from the point ones. Each trajectory has a globally unique ID, and its driving distance and driving time have been calculated.
- **Index Database:** This stores the previously mentioned inverted index.
- **Model Cache:** This stores the models that have been learned in the detection phase. In this way, when the same model is needed next time, it can be directly retrieved instead of being recalculated repeatedly. In addition, three-day expired time will be set to ensure the freshness of the model. Beyond this time, the model is retrained when the model is requested.



**Fig. 5.** Displacement, driving distance and driving time scatter plot. (Color figure online)

- **Anomalous Detection (STL):** For a testing trajectory, the historical trajectories that have the same source and destination point can be easily retrieved. Using the historical trajectories, our proposed anomalous detection method STL is applied to detect whether the testing trajectory is anomalous or not. The next section details the design of STL.

## 5 Design of STL

### 5.1 Basic Idea

Concerning transportation, driving distance and driving time are two important metrics to measure driving efficacy. We use the combination of these two metrics to detect trajectory anomaly which is more accurate and more efficient than the existing methods that separately utilize spatial or temporal features. When detecting a trajectory, if it incurs a reasonable cost in terms of driving distance or driving time to travel from the source to the destination, it should be considered as normal, no matter which routes it takes. Following this direction, we explore the detection of trajectory anomaly using two models that reveal the critical spatial-temporal laws: the D-S model characterizes the relationship between displacement (D) and driving distance (S) and the D-T model depicts the relationship between displacement and driving time (T). The basic idea behind STL is that the set of historical trajectories implying both D-S and D-T laws can be used to efficiently and precisely detect trajectory anomaly. By analyzing large number of historical trajectories, it is found that there are distinctive differences in both laws between anomalous trajectories and normal ones, as shown in Fig. 5. In Fig. 5a, the cyan lines represent the normal trajectories and the red line is an anomalous one. In Fig. 5b and c, the green points and the red points are extracted from the normal and anomalous trajectories respectively. As we can see, the D-S and D-T relationships are quite different after 10 km, so this part of the trajectory should be considered as anomalous.

Therefore, it is practical to use the D-S and D-T relationships of normal trajectories to train models that are, in turn, utilized to detect anomalous trajectories. Given a testing trajectory, for each of its point  $p_i$ , the displacement  $d_i$ ,

driving distance  $s_i$  and driving time  $t_i$  are first calculated, and then the models learned from the trajectories are utilized to predict the normal ranges for the driving distance and driving time of  $p_i$ . If both the real driving distance and driving time are beyond the predicted ranges,  $p_i$  is considered as anomalous, otherwise normal.

Therefore, designing STL concerns the following questions: (1) How to choose some trajectories from the taxi traces for training the two models? (2) How to learn the D-S and D-T models from the chosen trajectories? (3) How to perform trajectory detection based on the learned models? The following subsections answer these questions.

## 5.2 Selecting the Training Set

Since the historical trajectories stored in the database are unlabeled, a selection condition should be defined to retrieve some appropriate trajectories from the database to train the D-S and D-T models when detecting a testing trajectory. Therefore, an adaptive approach is firstly used to determine the appropriate number of historical trajectories in order to efficiently learn D-S and D-T models, and then a greedy strategy is applied to select the trajectories. Specifically, we define the *number of trajectories* as a function related to displacement ( $d$ ) between the source and destination point and the time of the source point ( $t$ ), i.e.,

$$m(d, t) = m_{base} \cdot 2^{1/(1+e^{-\gamma(d-d_{base})+wt(t)})} \quad (1)$$

where the  $m_{base}$  is the base number of trajectories,  $d_{base}$  is the base displacement, and  $wt(t)$  is the indicator of working hours (return 1 when at working hours, otherwise return 0). The intuition behind function (1) is that long-distance drives or drives occurring at working hours need more historical trajectories to keep the detection results stable.

To select the training set that has  $m(d, t)$  trajectories determined above, the trajectories that passed from the grid-cell of source point to the grid-cell of destination point are firstly extracted from the trajectory database. Since D-S and D-T models are time-sensitive, the historical trajectories whose timestamps of the source points are within a one-hour period either before or after the timestamp of the testing trajectory's source point and which have the same  $wt(t)$  attribute are chosen. If the number of currently selected trajectories is less than  $m(d, t)$ , the trajectories passing the source and destination nearby will be used as a supplement; otherwise, the  $m(d, t)$  trajectories closest to time  $t$  will be selected. Ultimately, to prevent anomalous trajectories from being selected, those trajectories whose driving distance or driving time are less than the median of all trajectories are selected as the training dataset.

### 5.3 Learning Models

As shown in Fig. 5, the relationships of D-S and D-T are not hard to fit by the linear model. The two models of STL are defined as follows:

$$\begin{aligned}
 s_i &= f(\mathbf{d}_i; \mathbf{w}_s) + \epsilon_s, \epsilon_s \sim \mathcal{N}(0, \sigma_s^2) \\
 t_i &= f(\mathbf{d}_i; \mathbf{w}_t) + \epsilon_t, \epsilon_t \sim \mathcal{N}(0, \sigma_t^2)
 \end{aligned}$$

where  $\mathbf{d}_i = [1, d_i, d_i^2, \dots]^T$ ,  $s_i$  and  $t_i$  represent the displacement vector, driving distance and driving time, respectively, of the  $i^{th}$  point in the training dataset, and  $f(\mathbf{d}_i; \mathbf{w}) = w_0 + w_1 d_i + w_2 d_i^2 + w_3 d_i^3 + \dots$ ,  $\mathbf{w} = [w_0, w_1, \dots]^T$ .  $\epsilon_s$  and  $\epsilon_t$  are random variables obeying the normal distribution. Therefore, when using the displacement  $\mathbf{d}_{test}$  of a test point to predict the driving distance and the driving time of this test point based on the learned models, we get two random variables subject to  $\mathcal{N}(f(\mathbf{d}_{test}; \mathbf{w}_s), \sigma_s^2)$  and  $\mathcal{N}(f(\mathbf{d}_{test}; \mathbf{w}_t), \sigma_t^2)$  which are useful to judge whether this point is anomalous or not. In the above models,  $\mathbf{w}_s$ ,  $\mathbf{w}_t$ ,  $\sigma_s$  and  $\sigma_t$  are the unknown parameters that need to be determined. We use the maximum likelihood estimation [8] to fit the models.

Applying this idea to STL, for each  $\mathbf{d}_i$  in the training dataset, the probability of  $s_i$  and  $t_i$  in the corresponding training dataset should be maximized, namely the following likelihood functions (2) and (3) should be maximized.

$$L_s = p(s_1, \dots, s_N | \mathbf{d}_1, \dots, \mathbf{d}_N, \mathbf{w}_s, \sigma_s^2) = \prod_{i=1}^N \mathcal{N}(\mathbf{w}_s^T \mathbf{d}_i, \sigma_s^2) \tag{2}$$

$$L_t = p(t_1, \dots, t_N | \mathbf{d}_1, \dots, \mathbf{d}_N, \mathbf{w}_t, \sigma_t^2) = \prod_{i=1}^N \mathcal{N}(\mathbf{w}_t^T \mathbf{d}_i, \sigma_t^2) \tag{3}$$

where  $N$  is the total amount of points in the training set.

In order to maximize the likelihood functions (2) and (3), the partial derivatives of  $\mathbf{w}$  and  $\sigma$  are respectively calculated for the log-likelihood function  $\log L$  as follows:

$$\begin{cases}
 \frac{\partial \log L_s}{\partial \mathbf{w}_s} = \frac{1}{\sigma_s^2} (\mathbf{D}^T \mathbf{s} - \mathbf{D}^T \mathbf{D} \mathbf{w}_s) = 0 \\
 \frac{\partial \log L_s}{\partial \sigma_s} = -\frac{N}{\sigma_s} + \frac{1}{\sigma_s^3} \sum_{i=1}^N (s_i - \mathbf{d}_i^T \mathbf{w}_s)^2 = 0 \\
 \frac{\partial \log L_t}{\partial \mathbf{w}_t} = \frac{1}{\sigma_t^2} (\mathbf{D}^T \mathbf{t} - \mathbf{D}^T \mathbf{D} \mathbf{w}_t) = 0 \\
 \frac{\partial \log L_t}{\partial \sigma_t} = -\frac{N}{\sigma_t} + \frac{1}{\sigma_t^3} \sum_{i=1}^N (t_i - \mathbf{d}_i^T \mathbf{w}_t)^2 = 0
 \end{cases}$$

By solving these equations, the results obtained are:

$$\begin{cases}
 \mathbf{w}_s = (\mathbf{D}^T \mathbf{D})^{-1} \mathbf{D}^T \mathbf{s} \\
 \sigma_s^2 = \frac{1}{N} (\mathbf{s}^T \mathbf{s} - \mathbf{s}^T \mathbf{D} \mathbf{w}_s)
 \end{cases} \tag{4}$$

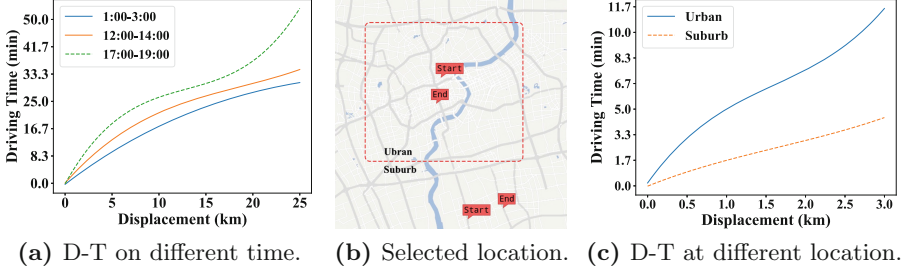


Fig. 6. Time-aware and location-aware.

$$\begin{cases} \mathbf{w}_t = (\mathbf{D}^T \mathbf{D})^{-1} \mathbf{D}^T \mathbf{t} \\ \sigma_t^2 = \frac{1}{N} (\mathbf{t}^T \mathbf{t} - \mathbf{t}^T \mathbf{D} \mathbf{w}_t) \end{cases} \quad (5)$$

where  $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_N]^T$ ,  $\mathbf{s} = [s_1, s_2, \dots, s_N]^T$  and  $\mathbf{t} = [t_1, t_2, \dots, t_N]^T$ .

When applying STL to detect the trajectories, it is apparent that STL has two main characteristics: time-aware and location-aware. Time-aware means the D-T model is changing at different times. Taking the source and destination shown in Fig. 5a for instance, the D-T model at different times is shown in Fig. 6a. It is intuitive that D-T has a smaller slope in the early hours of the morning (1:00–3:00) and a larger one during rush hour (17:00–19:00). With the same displacement, the driving time at rush hour is about 1.5 to 2 times more than that during the early hours of the morning, which proves that training different models based on time is reasonable. For location-aware, two source-destination pairs are picked from the urban and suburban area as shown in Fig. 6b. Figure 6c illustrates that the D-T model may have obvious differences at different locations, despite being at the same time.

#### 5.4 Performing Anomaly Detection

For any test point  $p_{test}$  in a testing trajectory, the  $\mathbf{d}_{test}$ ,  $s_{test}$  and  $t_{test}$  are firstly calculated. Secondly,  $\mathbf{d}_{test}$  is substituted into the D-S and D-T models that have been trained and two random variables are obtained: one for driving distance  $s$  subjected to  $\mathcal{N}(f(\mathbf{d}_{test}; \mathbf{w}_s), \sigma_s^2)$  and another for driving time  $t$  subjected to  $\mathcal{N}(f(\mathbf{d}_{test}; \mathbf{w}_t), \sigma_t^2)$ . For each distribution  $\mathcal{N}(\mu, \sigma^2)$  with probability density function  $f(x)$ , an anomaly probability function is defined as function (6):

$$anomaly(x) = \begin{cases} 0 & , x < \mu \\ 1 - f(x)/f(\mu) & , x \geq \mu \end{cases} \quad (6)$$

Therefore, two anomaly probabilities  $anomaly(s_{test})$  and  $anomaly(t_{test})$  are obtained. To combine these anomaly probabilities, the anomaly practicability of point  $p_{test}$  is defined as function (7):

$$anomaly(p_{test}) = \min(anomaly(s_{test}), anomaly(t_{test})) \quad (7)$$



**Table 1.** Detail of trace data

City	SH	SZ	CD
Data size	594 GB	32 GB	84 GB
# Taxis	13,585	9,475	14,424
# Trajectories	11,359,617	6,068,516	10,511,443
Sampling rate	10 s	10–30 s	10–30 s

where  $\min(\cdot, \cdot)$  is a function to return the smaller of two values. The principle under the combining strategy is that a taxi that drives a longer distance and spends more time is more likely to have detoured unnecessarily which corresponds to both higher  $anomaly(s_{test})$  and  $anomaly(t_{test})$ .

Anomaly probability  $anomaly(p_{test})$  represents the anomaly degree of one point in the trajectory. To evaluate the anomaly of the whole trajectory  $T$ , anomaly score based on the anomaly probability is defined as function (8):

$$score(T) = \sum_{i=1}^{N-1} dist(p_i, p_{i+1}) * anomaly(p_i) \quad (8)$$

where  $N$  is the number of points in trajectory  $T$ . For a point, larger anomaly probability with longer segment means higher anomaly in this trajectory.

## 6 Empirical Evaluation

### 6.1 Trace Data

The trace data used in the experiments were collected for a month in three cities of China, namely Shanghai (SH), Shenzhen (SZ) and Chengdu (CD). The detail of these data sets is summarized in Table 1, where the sampling rate means the interval between two adjacent GPS records. Both trace data and implementation of experiments are available at Github<sup>3</sup>.

To objectively and reasonably select the trajectory sets used for evaluation, the top 10 popular sources and destinations are selected from each data set and the source-destination pairs are identified. Each source-destination pair corresponds to a trajectory set in which all the trajectories pass the source and the destination. To make the performance more stable, those trajectory sets where the source and the destination are too close or the number of trajectories is less than 100 are ignored. The detail of the trajectory sets is listed in Table 2. In each trajectory set, the trajectory is labeled as anomalous or normal according to the taxi fare in this travel.

<sup>3</sup> <https://github.com/cbdog94/STL>.

**Table 2.** Detail of trajectory set

City	SH	SZ	CD
# Trajectory sets	62	48	65
Average # of trajectories	801.2	1051.1	3019.4
Min # of trajectories	112	151	236
Max # of trajectories	10,656	4,477	18,882

## 6.2 Evaluation Metrics

The TP rate (TPR) measures the proportion of anomalous trajectories that are correctly labeled and is defined as  $TPR = TP / (TP + FN)$ . The FP rate (FPR) measures the proportion of false alarms and is defined as  $FPR = FP / (FP + TN)$ .

## 6.3 Compared Methods

In the experiments, STL is compared with three detection methods, namely iBAT, iBOAT and OnATrade. iBAT [11] is an offline detection method based on the cell trajectory mapped from the GPS trajectory. To detect a testing trajectory, iBAT randomly picks a cell from the trajectory to split the trajectory set into two parts, the cell and exclude the cell respectively. This process is repeated until the testing trajectory is isolated or there are no more cells in the testing trajectory. iBOAT [2] is an online detection method which maintains an adaptive window containing the latest testing cells and compare the sub-trajectory composed of the cells in the window with historical trajectories. As long as the support ratio of the sub-trajectory is higher than a threshold  $\theta$ , the new cell will continue to be added to the window, otherwise, the adaptive window is reduced to include only the cell most recently added and this point is identified as anomalous. OnATrade [12] is an online detection method based on the digital map. OnATrade consists of two steps: route recommendation and online detection. If the maximum similarity between the testing trajectory and all the recommended trajectories is less than a threshold  $\theta$ , the testing point is considered as anomalous.

## 6.4 Experiment Setup

To hard-classify the anomaly of point in experiment, a threshold of anomaly probability calculated by function (7) is set up to 0.75. In addition,  $m_{base}$ ,  $d_{base}$  and  $\gamma$  in function (1) are set to 200, 20 km and 0.5 respectively. Similarly, the thresholds and parameters of iBAT, iBOAT and OnATrade are optimally set according to their proposed papers.

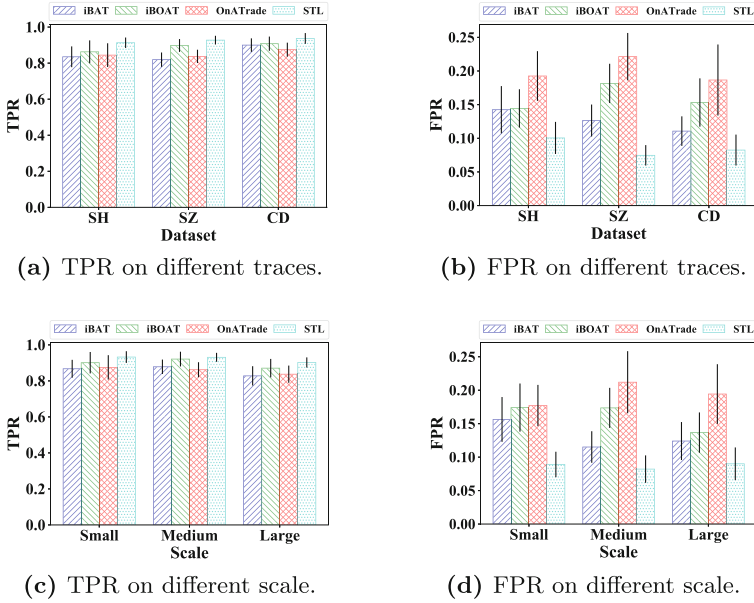


Fig. 7. Effectiveness comparison.

### 6.5 Experiment Results

**Effectiveness Evaluation.** In each trajectory set, leave-one-out cross validation is applied for the evaluation. Specifically, one trajectory is selected to test the detection methods and the rest in the set are used as historical trajectories. The results of the experiments are shown in Fig. 7, where Fig. 7a and b represent the average TPR and FPR on different traces. Compared with the other methods, the overall average of TPR of iBAT, iBOAT, OnATrade and STL are 85.4%, 89.9%, 85.3% and 92.5% respectively; and the overall average of FPR of these methods are 9.9%, 10.2%, 10.1% and 5.7% respectively. Compared to the other three methods, STL shows a slightly improvement over TPR and a substantial improvement over FPR by up to 43%. To evaluate the effectiveness on different scales of trajectory sets, the trajectory sets are divided into three parts: small (<300), medium (300–1000) and large (>1000) according to the number of trajectories in each set. As shown in Fig. 7c and d, when the scale of the trajectory set is small, STL still has good performance on FPR with 42.9%, 48.9% and 49.7% improvement over the other three methods, which demonstrates that STL does not need a large number of historical trajectories to perform the detection.

**Performance Evaluation.** We analyze the time cost for these detection methods. The average detection time per trajectory on different trace datasets is shown in Fig. 8. STL has the smallest time complexity while OnATrade has bad performance on a long-distance trajectory because the time complexity of

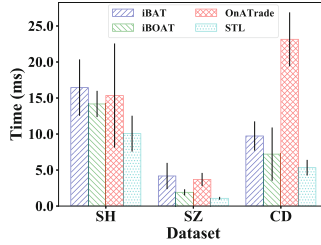


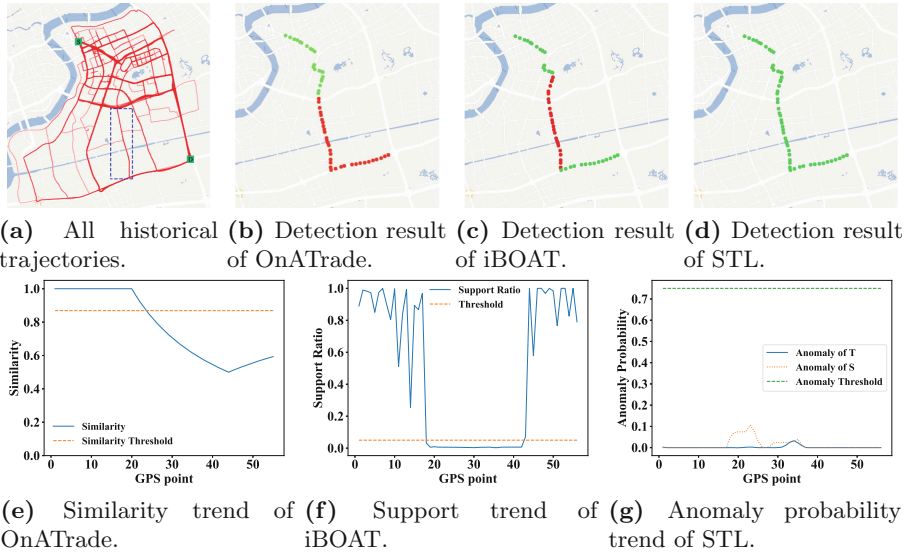
Fig. 8. Time cost comparison.

computing similarity is  $O(n^2)$  ( $n$  is the number of points in the trajectory). Since the GPS sampling rate on the data sets of Shenzhen and Chengdu is less than the one of Shanghai, the time costs on the two datasets are lower than the one on Shanghai on average. In addition, the space complexity of STL is  $O(1)$  because it only stores basic information, such as the source point, the last point and the driving distance of the last point. This is another advantage of STL over the other three methods.

**Online Detection Analysis.** This part analyzes why the FPR of iBOAT and OnATrade is higher than that of STL. To illustrate the difference between iBOAT, OnATrade and STL, one trajectory set consisting of 1490 trajectories from the Shanghai trace dataset is selected as an example as shown in Fig. 9a. The thicker the lines, the more times the trajectories are passed. The median of the driving distance and driving time of all trajectories are 13.56 km and 19.38 min, respectively. Since the number of trajectories covered by the rectangle shown by the blue dashed line is small, the trajectories passing this part are more likely to be classified as anomalous by iBOAT and OnATrade. We pick up a trajectory  $T$  passing this covered area, and its driving distance and driving time are 13.61 Km and 14.32 min, respectively. Since the driving time is less than the median and the driving distance is close to the median,  $T$  should be detected as normal.

Specifically, the visualization of the detection results for OnATrade and the similarity trend of  $T$  are shown in Fig. 9b and e. In Fig. 9b, the green points are labeled as normal and the red points are detected as anomalous. In Fig. 9e, the blue line and the orange dashed line represent the similarity of  $T$  with the recommended trajectories and the threshold  $\theta_{sim}$ , respectively. Since there is no trajectory in the recommended trajectories passing the covered area, it can be assumed that this part will be detected as abnormal. The detection result confirms this conjecture. The similarity drops from the 18<sup>th</sup> point and is below the threshold at the 23<sup>rd</sup> point. Therefore, after that, the points of  $T$  will be detected as anomalous.

For iBOAT, the visualization of the detection results is shown in Fig. 9c and the trend of the support ratio of  $T$  is shown in Fig. 9f. In Fig. 9f, the blue line and the orange dashed line represent the support ratio of each point in



**Fig. 9.** Online detection comparison. (Color figure online)

$T$  and the threshold  $\theta_{iBOAT}$ , respectively. The support ratio drops at the 18<sup>th</sup> point and rises at the 42<sup>nd</sup> point, so this part of  $T$  is detected as anomalous by iBOAT, which corresponds to the area covered by the blue dashed line rectangle in Fig. 9a.

For STL, the visualization of the detection results for STL is shown in Fig. 9d. In Fig. 9g, the blue line, orange dotted line and green dashed line represent the anomaly probability of driving time, driving distance and the threshold mentioned in Sect. 6.4, respectively. Since the two anomaly probabilities are always below the threshold, the anomaly probability of points are apparently within the normal range. Therefore,  $T$  is correctly detected as normal.

In addition, the anomaly scores of  $T$  are 0.59, 5422.90 and 37.08 for OnA-Trade, iBOAT and STL respectively, where the range of anomaly score for OnA-Trade is 0 to 1 and the range for iBOAT and STL is 0 to infinity. The anomaly scores of  $T$  are ranked 80 (5.4%), 50 (3.4%) and 926 (62.1%) in 1490 trajectories, which also reflects the different behavior between the three methods.

## 7 Conclusion

In this paper, we proposed a novel method called STL for the online detection of taxi trajectory anomaly. Rather than using spatial or temporal features separately, STL combines spatial features and temporal features to solve the problem of high false positives caused by most existing counting-based methods and to demand sufficient historical trajectories to work properly. STL catches the essence of transportation in terms of two critical models: D-S and D-T, which

can be used to detect anomalous trajectories efficiently and precisely. Through extensive experiments on real traces, we verify that STL can perform real-time detection and identify the anomalous part of trajectories. Furthermore, STL has excellent performance on both TPR and FPR with small time cost and memory consumption.


**Acknowledgments.** This work was supported by National Key R&D Program of China (2018YFB1003800), the National Science Foundation of China (61772334, 61702151, 61572324), the Joint Key Project of the National Natural Science Foundation of China (U1736207), and Shanghai Talent Development Fund, Shanghai Jiao Tong arts and science inter-project (15JCMY08).

## References

1. Chen, C., Zhang, D., Castro, P.S., Li, N., Sun, L., Li, S.: Real-time detection of anomalous taxi trajectories from GPS traces. In: Puiatti, A., Gu, T. (eds.) *MobiQuitous 2011*. LNCS, vol. 104, pp. 63–74. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-30973-1\\_6](https://doi.org/10.1007/978-3-642-30973-1_6)
2. Chen, C., et al.: iBOAT: Isolation-based online anomalous trajectory detection. *IEEE Trans. Intell. Transp. Syst.* **14**(2), 806–818 (2013)
3. Ge, Y., Xiong, H., Liu, C., Zhou, Z.: A taxi driving fraud detection system. In: *ICDM*, pp. 181–190 (2011)
4. Lee, J., Han, J., Li, X.: Trajectory outlier detection: a partition-and-detect framework. In: *ICDE*, pp. 140–149 (2008)
5. Li, X., Li, Z., Han, J., Lee, J.: Temporal outlier detection in vehicle traffic data. In: *ICDE*, pp. 1319–1322 (2009)
6. Liu, S., Ni, L.M., Krishnan, R.: Fraud detection from taxis' driving behaviors. *IEEE Trans. Veh. Technol.* **63**(1), 464–472 (2014)
7. Newson, P., Krumm, J.: Hidden Markov map matching through noise and sparseness. In: *ACM-GIS*, pp. 336–343 (2009)
8. Scholz, F.: Maximum likelihood estimation. In: *Encyclopedia of Statistical Sciences* (1985)
9. Wu, H., Sun, W., Zheng, B.: A fast trajectory outlier detection approach via driving behavior modeling. In: *CIKM*, pp. 837–846 (2017)
10. Yuan, J., et al.: T-drive: driving directions based on taxi trajectories. In: *ACM-GIS*, pp. 99–108 (2010)
11. Zhang, D., Li, N., Zhou, Z., Chen, C., Sun, L., Li, S.: iBAT: detecting anomalous taxi trajectories from GPS traces. In: *UbiComp*, pp. 99–108 (2011)
12. Zhou, Z., et al.: A method for real-time trajectory monitoring to improve taxi service using GPS big data. *Inf. Manage.* **53**(8), 964–977 (2016)
13. Zhu, J., Jiang, W., Liu, A., Liu, G.: Effective and efficient trajectory outlier detection based on time-dependent popular route. *World Wide Web* **20**(1), 111–134 (2017)
14. Zhu, J., Jiang, W., Liu, A., Liu, G., Zhao, L.: Time-dependent popular routes based trajectory outlier detection. In: Wang, J., et al. (eds.) *WISE 2015*. LNCS, vol. 9418, pp. 16–30. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-26190-4\\_2](https://doi.org/10.1007/978-3-319-26190-4_2)



# Correction to: Database Systems for Advanced Applications

Guoliang Li, Jun Yang, Joao Gama , Juggapong Natwichai,  
and Yongxin Tong

## Correction to:

**G. Li et al. (Eds.): *Database Systems for Advanced Applications*,  
LNCS 11447, <https://doi.org/10.1007/978-3-030-18579-4>**

In the original version of the chapter titled “An Exploration of Cross-Modal Retrieval for Unseen Concepts”, the acknowledgement was missing. It has been added.

In the original version of the chapter titled “Towards both Local and Global Query Result Diversification”, the funding information in the acknowledgement section was incomplete. This has now been corrected.

---

The updated version of these chapters can be found at  
[https://doi.org/10.1007/978-3-030-18579-4\\_2](https://doi.org/10.1007/978-3-030-18579-4_2)  
[https://doi.org/10.1007/978-3-030-18579-4\\_28](https://doi.org/10.1007/978-3-030-18579-4_28)

© Springer Nature Switzerland AG 2019  
G. Li et al. (Eds.): DASFAA 2019, LNCS 11447, p. C1, 2019.  
[https://doi.org/10.1007/978-3-030-18579-4\\_46](https://doi.org/10.1007/978-3-030-18579-4_46)

## Author Index

- An, Baoyi I-292, II-660  
Anirban, Shikha II-229  
Ao, Xiang II-555
- Bai, Mengna I-124  
Banerjee, Rohan I-625  
Bhowmick, Sourav S. I-350  
Bian, Bin-Bin I-70  
Boots, Robert I-587
- Cai, Peng I-209  
Cai, Tongzhao I-692  
Cai, Yi I-419, II-401  
Cao, Jialun II-70  
Cao, Jian II-764  
Cao, Xin I-398  
Cao, Zehong I-452  
Chang, Jia-Wei II-572  
Chao, Li II-156  
Chen, Chen I-571  
Chen, Chuan I-692  
Chen, Enhong I-795  
Chen, Guihai I-244, II-748  
Chen, Haiming II-70  
Chen, Jin I-709  
Chen, Jingxiao I-244  
Chen, Junsha II-730  
Chen, Kunjie II-713  
Chen, Man-Sheng I-175  
Chen, Shicheng I-260  
Chen, Wei I-676  
Chen, Weitong I-587, I-659, II-53, II-384  
Chen, Xia II-366  
Chen, Xiaojun II-485  
Chen, Xuanhao II-678  
Chen, Yubiao I-519  
Chen, Yun-chen II-433  
Chen, Yuting II-697  
Chen, Zhigang I-367  
Chen, Zhikui II-20  
Chen, Zitai I-692  
Cheng, Bin II-764  
Cheng, Huanyu II-464  
Chi, Jialin II-247
- Chung, Tae-Sun I-140  
Cochez, Michael I-659  
Cui, Bin II-139  
Cui, Pengjie II-536  
Cui, Zhiming II-301, II-317, II-333
- Dai, BingTian II-433  
Dayarathna, Miyuru II-264  
Decker, Stefan I-659  
Ding, Donghui II-366  
Ding, Zhuoye II-104  
Domeniconi, Carlotta I-313  
Du, Qing I-419  
Du, Yongjie I-19  
Du, Zihui I-19  
Duan, Lei I-107  
Duan, Yijun I-350  
Duan, Zhiqiang I-19
- Fan, Yi II-211  
Fang, Binxing I-501  
Fang, Junhua I-676, II-301, II-333  
Fang, Yixiang II-485  
Feng, Bo-Si I-70  
Feng, Dengguo II-247  
Feng, Jianhua I-725, I-760  
Feng, Shi I-468  
Fu, Ge I-3  
Fu, Hanjie I-536  
Fu, Shaojing I-227
- Gao, Chong II-70  
Gao, Chongming I-276  
Gao, Guoju I-292  
Gao, Hong I-742  
Gao, Neng II-730  
Gao, Xiaofeng I-244, II-748  
Gao, Yucen II-748  
Gu, Binbin I-676  
Gu, Yu II-626  
Guo, Chao II-485  
Guo, Guibing I-777  
Guo, Maozu I-313  
Guo, Xintong I-742



- Halford, Max II-3  
 Hamandawana, Prince I-140  
 Han, Chao I-107  
 Han, Dongxu II-121  
 Han, Jizhong I-36  
 Hao, Yuwei II-53  
 He, Fuzhen I-367  
 He, Liang II-173  
 He, Xiaofeng II-36  
 Hou, Yuexian II-366  
 Hu, Huiqi II-591  
 Hu, Songlin I-36  
 Hu, Wenxin II-173  
 Hua, Wen I-191  
 Huang, Dong I-175  
 Huang, Heyan I-70  
 Huang, Jen-Wei II-572  
 Huang, Joshua Zhexue II-191  
 Huang, Ling I-175  
 Huang, Yan I-725, I-760  
 Huang, Zi I-777  
  
 Indulska, Jadwiga II-384  
  
 Jatowt, Adam I-350  
 Jayasena, Sanath II-264  
 Jensen, Christian S. II-191, II-519  
 Jiang, Bo II-121  
 Jiang, Jiawei II-139  
 Jiang, Qize II-713  
 Jiang, Xiao-Jian I-70  
 Jiao, Pengfei I-435, II-502  
 Jin, Hai I-641  
 Jin, Peiquan II-642  
 Jin, Yuanyuan II-350  
  
 Kim, San I-725  
 Kong, Xiangnan I-484  
 Kwon, Se Jin I-140  
  
 Lai, Yongxuan II-211  
 Lan, Michael I-553  
 Latecki, Longin Jan II-211  
 Lei, Kai II-139  
 Lei, Xue II-401  
 Leung, Ho-fung II-401  
 Li, Binbin I-3  
 Li, Bohan I-587  
 Li, Chao I-3, II-86  
 Li, Chengqian II-211  
 Li, Faming I-519  
 Li, Hao II-247  
 Li, Haoda I-760  
 Li, Jiacheng I-36  
 Li, Jianxin II-464  
 Li, Jianzhong I-519  
 Li, Jiapeng I-676  
 Li, Jundong I-571, II-433  
 Li, Kaiyu I-725, I-760  
 Li, Liang I-52  
 Li, Minglu II-764  
 Li, Nan II-211  
 Li, Ning II-121, II-555  
 Li, Qi I-452  
 Li, Qing I-419, I-452, II-401  
 Li, Tianpeng II-502  
 Li, Xiuxing I-725  
 Li, Yeting II-70  
 Li, Yingshu I-519  
 Li, Yinuo I-641  
 Li, Yuan I-383  
 Li, Zepeng II-449  
 Li, Zhao II-366  
 Li, Zhixin II-555  
 Li, Zhixu I-260, I-367, I-676, II-317  
 Liang, Shen I-87  
 Liang, Shining II-53  
 Liao, Xiaofei I-641  
 Lin, Junyu I-36  
 Lin, Xuemin I-330, I-604  
 Lin, Zhangxi I-107  
 Ling, Guohui I-692  
 Liu, An I-260, I-292, I-367, II-660  
 Liu, Bing I-795  
 Liu, Chengfei II-609  
 Liu, Chenghao II-433  
 Liu, Chongzhi I-276  
 Liu, Guanfeng I-367  
 Liu, Hongtao I-435  
 Liu, Jian II-333  
 Liu, Jun I-659  
 Liu, Weidong II-104  
 Liu, Wenyan II-350  
 Liu, Yanchi II-301, II-317, II-333  
 Liu, Yaqiong I-398  
 Liu, Yong I-3  
 Lu, Chunyu I-435  
 Lu, Junwei I-244  
 Lu, Zhigang II-121  
 Luo, Anjing II-317

- Luo, Dexin II-191  
 Luo, Jiehuan I-398  
 Luo, Minnan I-571  
 Luo, Yuchuan I-227  
 Luo, Yuyu I-760  
 Lv, Guangyi I-795
- Ma, DeLong II-536  
 Ma, Huifang II-555  
 Ma, Jiangan I-87  
 Ma, Jingwei II-384  
 Ma, Yuliang II-536  
 Ma, Zongjie II-211  
 Mao, Rui II-485  
 Mao, Xian-Ling I-70  
 Mativenga, Ronnie I-140  
 Meng, Xiaofeng I-19  
 Min, Geyong II-20  
 Morvan, Franck II-3  
 Mu, Kedian I-383
- Nguyen, Quoc Viet Hung I-777  
 Nummenmaa, Jyrki I-107
- Pan, Kang I-760  
 Pan, Xuming II-366  
 Peng, Shunfeng II-697  
 Peng, Zhen I-571  
 Peng, Zhiyong II-281
- Qian, Shiyong II-764  
 Qian, Tieyun II-464  
 Qian, Weining I-209  
 Qian, Yu I-124  
 Qiang, Yang I-367  
 Qin, Ruiqi I-107  
 Qu, Qiang I-398
- Rajanala, Sailaja I-625  
 Ren, Da II-401  
 Ren, Haopeng I-419  
 Rodrigo, Aroscha II-264
- Saiful Islam, Md. II-229  
 Saint-Pierre, Philippe II-3  
 Shao, Junming I-276  
 Shao, Yingxia II-139  
 Shao, Zhiyuan I-641  
 Shen, Yanyan II-697
- Shen, Yuan II-281  
 Sheng, Victor S. II-301, II-317, II-333  
 Shi, Chengliang II-281  
 Shi, Zhenkun II-53  
 Singh, Manish I-625  
 Skoutas, Dimitrios I-553  
 Song, Jiaying II-104  
 Song, Wei II-281  
 Song, Yang II-173  
 Su, Kaile II-211  
 Su, Yuan I-501  
 Sun, Haobo II-139  
 Sun, Jianling II-433  
 Sun, Weiwei II-713  
 Sun, Xiaoxiao II-609  
 Sun, Xu I-157
- Tan, Zijiang I-157  
 Theodoratos, Dimitri I-553  
 Tian, Yingjie I-157  
 Tian, Yuan II-642  
 Tu, Chenyang II-730
- Wan, Shouhong II-642  
 Wang, Bo I-435, II-502  
 Wang, Chang-Dong I-175  
 Wang, Chen I-452  
 Wang, Daling I-468  
 Wang, Donghui I-209  
 Wang, Guoren I-52, II-536  
 Wang, Hengliang I-383  
 Wang, Hsiu-Yi II-572  
 Wang, Jiang II-139  
 Wang, Jingli I-36  
 Wang, Jun I-313  
 Wang, Junhu II-229  
 Wang, Liping I-330  
 Wang, Liwei I-191  
 Wang, Lu II-36  
 Wang, Meng I-659  
 Wang, Panpan II-366  
 Wang, Qinyong I-777  
 Wang, Ruijie I-659  
 Wang, Senzhang I-501  
 Wang, Shuai I-795  
 Wang, Shupeng I-3  
 Wang, Wei I-536  
 Wang, Weiqing I-777  
 Wang, Wenjun I-435, II-502

- Wang, Xiaoling I-709, II-350  
 Wang, Xinyu I-709  
 Wang, Yanda I-587  
 Wang, Yang I-244  
 Wang, Yaping I-435, II-502  
 Wang, Ying II-464  
 Wang, Yishu II-536  
 Wang, Yong I-3, I-760  
 Wang, Yuehui I-313  
 Wei, Xiaochi I-70  
 Wen, Jiahui II-384  
 Wu, Changfa II-626  
 Wu, Dingming II-191, II-519  
 Wu, Gang I-52  
 Wu, Guangjun I-3  
 Wu, Jianjun II-121  
 Wu, Jie II-660  
 Wu, Longcan I-468  
 Wu, Min II-433  
 Wu, Xiaoying I-553  
 Wu, Xudong I-604  
 Wu, Xunxun II-502  
 Wu, Yiyu II-609
- Xia, Jinfu II-416  
 Xia, Long II-104  
 Xiao, Mingjun I-292, II-660  
 Xie, Haoran I-419  
 Xing, Chunxiao II-86, II-156, II-591  
 Xiong, Yun I-484, I-536  
 Xu, Jiajie II-301, II-317, II-333  
 Xu, Jie II-156  
 Xu, Jingyun II-401  
 Xu, Ming I-227  
 Xu, Shaofeng I-484  
 Xu, Yu II-139  
 Xue, Cong II-730  
 Xue, Guangtao II-764
- Yang, Chen I-19  
 Yang, Huiping II-416  
 Yang, Junye II-591  
 Yang, Kejia I-157  
 Yang, Qinli I-276  
 Yang, Shiyu I-604  
 Yang, Songfan I-725, I-760  
 Yang, Weidong I-157  
 Yang, Xiaochen I-227  
 Yang, Yang I-330  
 Yao, Bin II-416
- Ye, Yutong II-247  
 Ye, Zixin I-191  
 Yi, Feng II-121  
 Yin, Dawei II-104  
 Yin, Hongzhi I-777  
 Yin, Jianwen II-433  
 Yoshikawa, Masatoshi I-350  
 Yu, Dongjin II-609  
 Yu, Dongxiao I-641  
 Yu, Ge I-468, II-626  
 Yu, Guoxian I-313  
 Yu, Jiadi II-764  
 Yu, Philip S. I-501, I-536  
 Yuan, Haitao I-760  
 Yuan, Hua I-124  
 Yuan, Jiahao II-350  
 Yuan, Long I-604  
 Yuan, Ye I-52, II-536  
 Yue, Lihua II-642  
 Yue, Lin II-53  
 Yun, Xiaochun I-3
- Zeng, ZeQuan I-419  
 Zha, Daren II-730  
 Zhai, Dongjun I-260  
 Zhang, Kun I-795  
 Zhang, Li II-449  
 Zhang, Min I-367, II-247, II-416  
 Zhang, Peng I-107  
 Zhang, Tao II-764  
 Zhang, Tianle I-501  
 Zhang, Tingting II-301  
 Zhang, Wei II-36  
 Zhang, Wenjie I-330, I-604  
 Zhang, Xi I-501  
 Zhang, Xiangliang I-260, I-313  
 Zhang, Xiaolan II-70  
 Zhang, Yan II-86  
 Zhang, Yanchun I-87  
 Zhang, Yao I-536  
 Zhang, Yifei I-468  
 Zhang, Ying I-330, I-709  
 Zhang, Yong II-86, II-156, II-591  
 Zhang, Zhong I-276  
 Zhang, Zhou II-642  
 Zhao, Chenfei I-383  
 Zhao, Hui I-292, II-660  
 Zhao, Kangzhi II-86  
 Zhao, Lei I-367, I-676, II-301, II-317, II-333  
 Zhao, Pengpeng I-367, II-301, II-317, II-333

- Zhao, Tianyu I-760  
Zhao, Weizhong II-555  
Zhao, Yan II-416, II-678  
Zhao, Zhihui I-3  
Zheng, Baihua II-713  
Zheng, Kai II-416, II-678  
Zheng, Qinghua I-571  
Zheng, Zibin I-692  
Zhong, Fangming II-20  
Zhong, Jiang I-452  
Zhong, Ming II-464  
Zhong, Mingyang II-384  
Zhou, Aoying I-209  
Zhou, Hao II-748  
Zhou, Jun II-211  
Zhou, Lianming II-678  
Zhou, Wei I-36  
Zhou, Xiangdong I-157  
Zhou, Xiaofang I-191, II-384  
Zhu, Fu I-760  
Zhu, Lin I-157  
Zhu, Yangyong I-484, I-536  
Zhu, Yanmin II-697, II-764  
Zhu, Yi II-519  
Zhu, Yuanyuan II-464  
Zou, Lixin II-104  
Zou, Zhaonian I-519, I-742  
Zuo, Wanli II-53