

Christine Strauss · Toshiyuki Amagasa ·
Giuseppe Manco · Gabriele Kotsis ·
A Min Tjoa · Ismail Khalil (Eds.)

LNCS 14910

Database and Expert Systems Applications

35th International Conference, DEXA 2024
Naples, Italy, August 26–28, 2024
Proceedings, Part I

1
Part I

DEXA 2024

 Springer

Lecture Notes in Computer Science

14910

Founding Editors

Gerhard Goos
Juris Hartmanis

Editorial Board Members

Elisa Bertino, *Purdue University, West Lafayette, IN, USA*

Wen Gao, *Peking University, Beijing, China*

Bernhard Steffen , *TU Dortmund University, Dortmund, Germany*

Moti Yung , *Columbia University, New York, NY, USA*

The series Lecture Notes in Computer Science (LNCS), including its subseries Lecture Notes in Artificial Intelligence (LNAI) and Lecture Notes in Bioinformatics (LNBI), has established itself as a medium for the publication of new developments in computer science and information technology research, teaching, and education.

LNCS enjoys close cooperation with the computer science R & D community, the series counts many renowned academics among its volume editors and paper authors, and collaborates with prestigious societies. Its mission is to serve this international community by providing an invaluable service, mainly focused on the publication of conference and workshop proceedings and postproceedings. LNCS commenced publication in 1973.


Christine Strauss · Toshiyuki Amagasa ·
Giuseppe Manco · Gabriele Kotsis · A Min Tjoa ·
Ismail Khalil
Editors

Database and Expert Systems Applications

35th International Conference, DEXA 2024
Naples, Italy, August 26–28, 2024
Proceedings, Part I

Editors

Christine Strauss
University of Vienna
Vienna, Austria

Toshiyuki Amagasa 
University of Tsukuba
Tsukuba, Japan

Giuseppe Manco
National Research Council (CNR)
Rende, Italy

Gabriele Kotsis
Johannes Kepler University Linz
Linz, Austria

A Min Tjoa 
Vienna University of Technology
Vienna, Austria

Ismail Khalil
Johannes Kepler University Linz
Linz, Austria

ISSN 0302-9743

ISSN 1611-3349 (electronic)

Lecture Notes in Computer Science

ISBN 978-3-031-68308-4

ISBN 978-3-031-68309-1 (eBook)

<https://doi.org/10.1007/978-3-031-68309-1>

© The Editor(s) (if applicable) and The Author(s), under exclusive license
to Springer Nature Switzerland AG 2024

This work is subject to copyright. All rights are solely and exclusively licensed 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

If disposing of this product, please recycle the paper.

Preface

We are pleased to present the proceedings of the DEXA 2024 conference, which showcased the latest advancements and research in database expert systems applications. This collection of papers represents the cutting-edge work of scholars and practitioners from around the world, exploring a diverse array of topics that highlight the crucial role of databases in advanced technology and modern society.

The DEXA 2024 proceedings consist of two volumes and include papers from the 35th DEXA edition, which took place on August 26–28, 2024 in Naples (Italy).

DEXA 2024 received a total of 102 paper submissions. From these, the Program Committee selected 27 as regular papers, resulting in an acceptance rate of 26%. Additionally, 20 papers were accepted as short papers to showcase pioneering research and innovative projects across various disciplines. These short papers highlight early-stage research, novel ideas, and preliminary findings, fostering meaningful discussions and potential collaborations.

The single-blind peer review process provided evaluations of each submitted manuscript by at least three reviewers. Their assessment not only served the purpose of quality control for the conference but also contained valuable comments for the authors and inspiring ideas for improvement or further research. We would like to sincerely thank our Program Committee members and external reviewers for their critical and motivating reviews.

Accepted papers cover a wide variety of research topics on both theoretical and practical aspects.

The papers on **Financial and Economic Data Analysis** present innovative approaches to financial policy retrieval, stock trading strategies using reinforcement learning, and comprehensive databases for network pharmacology research. These contributions reflect the growing intersection of finance, economics, and advanced data analytics.

Papers on **Graph Theory and Network Analysis** delve into fast subgraph search techniques, knowledge graph alignment, dynamic graph indexing, and enhanced navigable small world searches. The studies presented in these papers offer novel solutions for managing and interpreting large-scale graph data.

The section on **Database Management and Query Optimization** features research on improving text-to-SQL tools, deep learning-based workload encoding, hierarchical storage management, and top-k stabbing queries on weighted interval data. These contributions aim to enhance the performance and the accuracy of database systems.

The contributions on the topic of **Machine Learning and Large Language Models** explore pre-trained knowledge tracing models, machine learning on serverless platforms, performance evaluation of large language models, and robust table-to-text generation. These studies highlight the transformative impact of machine learning and data science on database applications.

The section on **Recommender Systems and Personalization** includes research on collaborative filtering for patient outcomes, sequential recommendation algorithms, context-aware recommendation systems, and category-aware sequential recommendations. These papers demonstrate the effectiveness of recommender systems in delivering tailored user experiences.

The papers on **Blockchain and Supply Chain Management** discuss integrating blockchain with encrypted-RSA NFTs and IPFS, as well as resource-oriented approaches for blockchain integration. These studies underscore the potential of blockchain to revolutionize supply chain management.

The section on **Data Mining and Knowledge Discovery** presents work on abnormal citation group detection, semantic word grouping for text classification, and knowledge graph management. These contributions represent advances in the field of data mining and enhance our ability to discover meaningful patterns and relationships.

The papers on **Spatiotemporal Data and Mobility Analysis** address efficient movelet extraction, human mobility analysis, traffic flow prediction, and spatio-temporal graph modeling. These studies provide innovative methods for analyzing and interpreting spatiotemporal data.

The section on **Computer Vision and Image Processing** features research on video situation monitoring, disaster image tagging, and segmentation of CNC milling sensor data. These papers highlight the advancements in visual data analysis and their practical applications.

The section on **Data Security and Privacy** includes studies on identifying personal identifiable information in unstructured text and designing wireless signal propagation models. These contributions address the critical challenges of maintaining data security and protecting user privacy.

The papers on **Database Indexing and Query Processing** explore diverse index tuning, fast learned cardinality estimation, diffusion pre-computation for image retrieval, and high contention management in B-trees. These studies offer innovative solutions to improve database indexing and query efficiency.

Finally, the section on **Specialized Applications and Case Studies** presents specific applications and case studies that demonstrate the practical implementation of advanced database technologies. Topics include epidemic prediction with spatiotemporal graph neural networks, virtual data lake zones, protein surface classification, and legal case retrieval representation. These papers provide valuable insights into the application of database systems in various fields.

We extend our heartfelt gratitude to our keynote speakers: Carlo Sansone from the University of Naples Federico II, Italy; A Min Tjoa from Vienna University of Technology, Austria; and Toshiyuki Amagasa from the University of Tsukuba, Japan, for their exceptional presentations at DEXA 2024 and its related conferences and workshops. Their insights were enlightening and deeply resonated with our participants, making their contributions one of the event's highlights.

Our gratitude goes to all the authors and participants who contributed to this conference. Their dedication and expertise made this event a success. We hope these proceedings inspire further research and innovation in the field of database expert systems applications.

August 2024

Christine Strauss
Toshiyuki Amagasa
Giuseppe Manco
Gabriele Kotsis
A Min Tjoa
Ismail Khalil

Organization

Program Committee Chairs

| | |
|-------------------|--|
| Christine Strauss | University of Vienna, Austria |
| Toshiyuki Amagasa | University of Tsukuba, Japan |
| Giuseppe Manco | Italian National Research Council (CNR), Italy |

Steering Committee

| | |
|-----------------|---|
| Gabriele Kotsis | Johannes Kepler University Linz, Austria |
| A Min Tjoa | Vienna University of Technology, Austria |
| Lukas Fischer | Software Competence Center Hagenberg, Austria |
| Bernhard Moser | Software Competence Center Hagenberg, Austria |
| Ismail Khalil | Johannes Kepler University Linz, Austria |
| Nicola Mazzocca | University of Naples Federico II, Italy |
| Elio Masciari | University of Naples Federico II, Italy |

Program Committee Members

| | |
|--------------------|--|
| A Min Tjoa | Vienna University of Technology, Austria |
| Abdessamad Imine | Loria, France |
| Allel Hadjali | LIAS/ENSMA, France |
| Andreas Ekelhart | Secure Business Austria, Austria |
| Anne Kayem | University of Exeter, UK |
| Athman Bouguettaya | University of Sydney, Australia |
| Bala Srinivasan | Monash University, Australia |
| Bettina Fazzinga | University of Calabria, Italy |
| Brahim Ouhbi | ENSAM, Morocco |
| Cedric Du Mouza | CNAM, France |
| Christian Stummer | Bielefeld University, Germany |
| Chuan-Ming Liu | National Taipei University of Technology, Taiwan |
| Claudia Roncancio | Grenoble University, France |
| Daisuke Kitayama | Kogakuin University, Japan |
| Deborah Dahl | Conversational Technologies, USA |
| Elio Masciari | University of Naples Federico II, Italy |

| | |
|-------------------------|---|
| Elmar Kiesling | Vienna University of Economics and Business, Austria |
| Ettore Ritacco | University of Udine, Italy |
| Flavio Ferrarotti | Software Competence Centre Hagenberg, Austria |
| Flavius FrasinCAR | Erasmus University Rotterdam, The Netherlands |
| Florence Sedes | IRIT, Université Toulouse III Paul Sabatier, France |
| Francesc D. Muñoz-Escof | Universitat Politècnica de València, Spain |
| Franck Morvan | IRIT - Université Toulouse III Paul Sabatier, France |
| Gheorghe Cosmin Silaghi | Babeş-Bolyai University, Romania |
| Giovanna Guerrini | University of Genova, Italy |
| Hamidah Ibrahim | Universiti Putra Malaysia, Malaysia |
| Hiroyuki Toda | Yokohama City University, Japan |
| Idir Amine Amarouche | USTHB, Algeria |
| Iker Pastor-López | University of Deusto, Spain |
| Ionut Iacob | Georgia Southern University, USA |
| Ismael Navas-Delgado | University of Málaga, Spain |
| Ivan Izonin | Lviv Polytechnic National University, Ukraine |
| Ivanna Dronyuk | Jan Dlugosz University, Poland |
| Javier Nieves | Azterlan, Spain |
| Jérôme Darmont | Université Lyon 2, France |
| Jianwei Zhang | Iwate University, Japan |
| Jorge Lloret | University of Zaragoza, Spain |
| Josef Küng | Johannes Kepler Universität Linz, Austria |
| Jun Miyazaki | Tokyo Institute of Technology, Japan |
| Karim Benouaret | Université Claude Bernard Lyon 1, France |
| Lars Mönch | University of Hagen, Germany |
| Lenka Lhotska | CVUT, Czech Republic |
| Louise Parkin | LIAS, France |
| Luca Caviglione | IMATI-CNR, France |
| Manfred Hauswirth | TU Berlin, Germany |
| Manolis Gergatsoulis | Ionian University, Greece |
| Marcin Paprzycki | IBS PAN and WSM, Poland |
| Marcin Skowron | Deepsearch GmbH, Austria |
| Marinette Savonnet | University of Burgundy, France |
| Markus Endres | University of Passau, Germany |
| Massimo Ruffolo | ICAR-CNR, Italy |
| Massimo Guarascio | ICAR-CNR, Italy |
| Michael Sheng | Macquarie University, Australia |
| Michal Kratky | VSB-Technical University of Ostrava, Czech Republic |

| | |
|---------------------|--|
| Mizuho Iwaihara | Waseda University, Japan |
| Mustafa Atay | Winston-Salem State University, USA |
| Nora Faci | Université Lyon 1, France |
| Olivier Teste | IRIT, France |
| Omar Boussaid | ERIC Laboratory, France |
| Pavlo Radiuk | Khmelnytskyi National University, Ukraine |
| Peiquan Jin | University of Science and Technology of China, China |
| Petr Kremen | Czech Technical University in Prague, Czech Republic |
| Qiang Ma | Kyoto University, Japan |
| Qiang Zhu | University of Michigan - Dearborn, USA |
| Rachid Anane | Coventry University, UK |
| Riad Mokadem | Paul Sabatier University, France |
| Riccardo Albertoni | CNR-IMATI, Italy |
| Sergio Ilarri | University of Zaragoza, Spain |
| Simone Raponi | NATO STO CMRE, Italy |
| Soon Chun | City University of New York, USA |
| Soumyava Das | Teradata Labs, USA |
| Srinath Srinivasa | International Institute of Information Technology, Bangalore, India |
| Stephane Bressan | National University of Singapore, Singapore |
| Stephane Jean | University of Poitiers, ISAE-ENSMA, LIAS, France |
| Sven Hartmann | Clausthal University of Technology, German |
| Sven Groppe | University of Lübeck, Germany |
| Traian Marius Truta | Northern Kentucky University, USA |
| Vicenc Torra | University of Skövde, Sweden |
| Vincenzo Deufemia | University of Salerno, Italy |
| Vitaliy Yakovyna | Lviv Polytechnic National University, Ukraine |
| Wilfrid Utz | OMiLAB gGmbH, Germany |
| Yan Zhu | Southwest Jiaotong University, China |
| Yang-Sae Moon | Kangwon National University, South Korea |
| Yousuke Watanabe | Nagoya University, Japan |

External Reviewers

| | |
|-------------------------|-----------------------------------|
| A. K. M. Tauhidul Islam | Informatica, USA |
| Alexander Völz | University of Vienna, Austria |
| Amira Barhoumi | Grenoble Alpes University, France |
| Angelica Liguori | ICAR-CNR, Italy |

Babar Shahzaad
Bing Huang
Chuan-Chi Lai
Gaetano Cimino
Gianluigi Folino
Muhammad Umair
Simone Mungari

University of Sydney, Australia
Nanyang Technological University, Singapore
National Chung Cheng University, Taiwan
Università degli Studi di Salerno, Italy
ICAR-CNR, Italy
University of Sydney, Australia
ICAR-CNR, Italy

Organizers



Abstracts of Keynote Talks

Multimodal Deep Learning in Medical Imaging

Carlo Sansone

Department of Electrical Engineering and Information Technology,
University of Naples Federico II, Italy

Abstract. In this talk, we will consider how Deep Learning (DL) approaches can profitably exploit the presence of multiple data sources in the medical domain.

First, the need to be able to use information from multimodal data sources is addressed. Starting from an analysis of different multimodal data fusion techniques, an innovative approach will be proposed that allows the different modalities to influence each other.

However, in medical applications it is often very difficult to obtain high quality and balanced labelled datasets due to privacy and sharing policy issues. Therefore, several applications have leveraged DL approaches in data augmentation techniques, proposing models that can create new realistic and synthetic samples. Consequently, a new data source can be identified, namely a synthetic data source. In this context, a data augmentation method based on deep learning, specifically designed for the medical domain, will be presented. It exploits the biological characteristics of images by implementing a physiologically-aware synthetic image generation process.

Digital Humanism as an Enabler for a Holistic Socio-Technical Approach to the Latest Developments in Computer Science and Artificial Intelligence

A Min Tjoa

TU Wien (Vienna University of Technology), Austria

Abstract. The rapid development of computer science and artificial intelligence (AI) has brought about transformative changes, but not without significant ethical, social, and technical challenges. As early as 2017, Tim Berners-Lee, the inventor of the World Wide Web, warned of the “nasty storm” threatening the future of the web, including the proliferation of fake news, propaganda, and increasing polarization. These issues highlight the urgent need for a paradigm that ensures technology serves the best interests of humanity.

This keynote will explore the foundational principles of Digital Humanism and its role in guiding the development of computer science and AI to align with human values and societal well-being.

In December 2023, the United Nations Advisory Panel on AI released its interim report, “Governing AI for Humanity,” which highlights the need for AI governance to address challenges and harness AI’s potential in an inclusive way, ensuring that no one is left behind. A key measure of AI’s success will be its contribution to achieving the SDGs.

The keynote will illustrate how Digital Humanism can be operationalized to create technologies that enhance human capabilities and societal well-being. It will highlight the need for interdisciplinary research and development to harness the potential of computer science and AI for the benefit of humanity.

Digital Humanism offers a vital pathway for navigating the complexities of modern technological advancements. By taking a holistic socio-technical approach, it can be ensured that developments in computer science and AI are aligned with our core human values, thereby fostering a more just, ethical, and sustainable digital future.

Deep Entity Processing in the Era of Large Language Models: Challenges and Opportunities

Toshiyuki Amagasa

University of Tsukuba, Japan

Abstract. Handling entities has long been a critical task in data analytics and integration, with over 80% of time and effort often devoted to data pre-processing. Improving the performance of this task has been a persistent challenge. Recently, transformer-based pre-trained language models and large language models (LLMs) have emerged as key tools for entity processing tasks such as named entity recognition (NER) and entity matching. However, these models introduce new challenges, including significant demands for computational resources and high-quality training data. In this talk, we will review recent advances in deep entity processing and explore the associated challenges and research opportunities.

Contents – Part I

Financial and Economic Data Analysis

| | |
|--|----|
| CSPRD: A Financial Policy Retrieval Dataset for Chinese Stock Market | 3 |
| <i>Jinyuan Wang, Zhong Wang, Zeyang Zhu, Jinhao Xie, Yong Yu, Yongjian Fei, Yue Huang, Dawei Cheng, and Hai Zhao</i> | |
| Leveraging Heterogeneous Text Data for Reinforcement Learning-Based Stock Trading Strategies | 18 |
| <i>Keishi Fukuda and Qiang Ma</i> | |
| TCMIDP: A Comprehensive Database of Traditional Chinese Medicine for Network Pharmacology Research | 34 |
| <i>Lei Ye, Wenxiang Liu, Yuhao Zheng, and Jianhua Li</i> | |

Graph Theory and Network Analysis

| | |
|---|----|
| Fast Subgraph Search with Graph Code Indices | 43 |
| <i>Naoya Funamoto and Akihiro Inokuchi</i> | |
| Completing Predicates Based on Alignment Rules from Knowledge Graphs | 59 |
| <i>Emetis Niazmand and Maria-Esther Vidal</i> | |
| Enriching Hierarchical Navigable Small World Searches with Result Diversification | 75 |
| <i>Mauro Weber, João Silva-Leite, Lúcio F. D. Santos, Daniel de Oliveira, and Marcos Bedo</i> | |
| An Efficient Indexing Method for Dynamic Graph k NN | 81 |
| <i>Shohei Matsugu, Suomi Kobayashi, and Hiroaki Shiokawa</i> | |

Database Management and Query Optimization

| | |
|--|----|
| Improving the Accuracy of Text-to-SQL Tools Based on Large Language Models for Real-World Relational Databases | 93 |
| <i>Gustavo M. C. Coelho, Eduardo R. S. Nascimento, Yenier T. Izquierdo, Grettel M. García, Lucas Feijó, Melissa Lemos, Robinson L. S. Garcia, Aiko R. de Oliveira, João P. Pinheiro, and Marco A. Casanova</i> | |

| | |
|---|-----|
| QPSEncoder: A Database Workload Encoder with Deep Learning | 108 |
| <i>Jianwen Yang, QiuHong Zhang, Jin Yan, Zhiming Ding, Meiling Zhu, and Xinjie Lv</i> | |
| Efficient Random Sampling from Very Large Databases | 124 |
| <i>Idan Cohen, Aviv Yehezkel, and Zohar Yakhini</i> | |
| SQL-to-Schema Enhances Schema Linking in Text-to-SQL | 139 |
| <i>Sun Yang, Qiong Su, Zhishuai Li, Ziyue Li, Hangyu Mao, Chenxi Liu, and Rui Zhao</i> | |
| Efficient Algorithms for Top-k Stabbing Queries on Weighted Interval Data | 146 |
| <i>Daichi Amagata, Junya Yamada, Yuchen Ji, and Takahiro Hara</i> | |
| A Hierarchical Storage Mechanism for Hot and Cold Data Based on Temperature Model | 153 |
| <i>Shicong Ma, Tianhai Zhao, Jianhua Gu, and Yunlan Wang</i> | |
| Machine Learning and Large Language Models | |
| A Pre-trained Knowledge Tracing Model with Limited Data | 163 |
| <i>Wenli Yue, Wei Su, Lei Liu, Chuan Cai, Yongna Yuan, Zhongfeng Jia, Jiamin Liu, and Wenjian Xie</i> | |
| Chorus: More Efficient Machine Learning on Serverless Platform | 179 |
| <i>Guang Yang, Jie Liu, Shuai Wang, Muzi Qu, Dan Ye, and Hua Zhong</i> | |
| Evaluating Performance of LLaMA2 Large Language Model Enhanced by QLoRA Fine-Tuning for English Grammatical Error Correction | 194 |
| <i>Jing An, Yanbing Bai, Jiyi Li, Junjie Hu, Rui Li, Yuxi Xiao, and Rui Hua</i> | |
| A Label Embedding Algorithm Based on Maximizing Normalized Cross-Covariance Operator | 207 |
| <i>Yulin Xue, Yuchen Pan, Tao Peng, Jun Li, and Jianhua Xu</i> | |
| Analyzing the Efficacy of Large Language Models: A Comparative Study | 215 |
| <i>Sonia Khetarpaul, Dolly Sharma, Shreya Sinha, Aryan Nagpal, and Aarush Narang</i> | |
| Leveraging Large Language Models for Flexible and Robust Table-to-Text Generation | 222 |
| <i>Ermelinda Oro, Luca De Grandis, Francesco Maria Granata, and Massimo Ruffolo</i> | |

Recommender Systems and Personalization

Collaborative Filtering for the Imputation of Patient Reported Outcomes 231
*Eric Ababio Anyimadu, Clifton David Fuller, Xinhua Zhang,
 G. Elisabeta Marai, and Guadalupe Canahuate*

**Category-Aware Sequential Recommendation with Time Intervals
of Purchases** 249
Jia-Ling Koh and Cheng-Wei Chen

A Soft Actor-Critic Algorithm for Sequential Recommendation 258
Hyejin Hong, Yusuke Kimura, and Kenji Hatano

**An Approach for Social-Distance Preserving Location-Aware
Recommender Systems: A Use Case in a Hospital Environment** 267
*Marcos Caballero, María del Carmen Rodríguez-Hernández,
Raúl Parada, Sergio Ilarri, Raquel Trillo-Lado, Ramón Hermoso,
and Óscar J. Rubio*

Author Index 275

Contents – Part II

Blockchain and Supply Chain Management

| | |
|---|---|
| Towards Transparent and Ethical Coffee and Cocoa Supply Chains: Integrating Blockchain, Encrypted-RSA NFTs, and IPFS | 3 |
| <i>P. H. T. Trung, L. K. Bang, H. N. Kha, Q. T. Bao, N. D. P. Trong, V. C. P. Loc, D. M. Hieu, and D. T. Khoa</i> | |

| | |
|---|----|
| Resource-Oriented Approach for Effective Blockchain Integration in Intertwined Supply Chains | 18 |
| <i>Devis Bianchini, Valeria De Antonellis, Massimiliano Garda, and Michele Melchiori</i> | |

Data Mining and Knowledge Discovery

| | |
|--|----|
| GADEC: Discovering Abnormal Citation Groups Based on Enhanced Local Community Expansion and DQN | 37 |
| <i>Yan Zhu, Xiaofei Wang, Xinrui Lin, and Yiqiang Peng</i> | |

| | |
|---|----|
| Knowledge Graph Creation and Management Made Easy with KGraphX | 53 |
| <i>Ahmad Hemid, Abderrahmane Khiat, Megha Jayakumar, Christoph Lange, Christoph Quix, and Stefan Decker</i> | |

| | |
|---|----|
| SEMANT - Feature Group Selection Utilizing FastText-Based Semantic Word Grouping, Scoring, and Modeling Approach for Text Classification | 69 |
| <i>Daniel Voskergian, Burcu Bakir-Gungor, and Malik Yousef</i> | |

Spatiotemporal Data and Mobility Analysis

| | |
|---|----|
| UltraMovelets: Efficient Movelet Extraction for Multiple Aspect Trajectory Classification | 79 |
| <i>Tarlis Tortelli Portela, Vanessa Lago Machado, Jonata Tyska Carvalho, Vania Bogorny, Anna Bernasconi, and Chiara Renso</i> | |

| | |
|--|----|
| Understanding Human Mobility Characteristics Through Behavior and Corresponding Environmental Information | 95 |
| <i>Ryuichi Sudo and Hiroyuki Toda</i> | |

| | |
|--|-----|
| Traffic Flow Prediction Based on Deep Spatio-Temporal Domain Adaptation | 110 |
| <i>Zhihui Wang and Bingxin Li</i> | |
| Exploring of STGNN for Traffic Forecasting at Expanding Traffic Network | 116 |
| <i>Tomoki Kawabata and Hiroyuki Toda</i> | |
| Computer Vision and Image Processing | |
| Video Situation Monitoring Using Continuous Queries | 125 |
| <i>Hafsa Billah and Sharma Chakravarthy</i> | |
| Semi-automated Disaster Image Tagging While Protecting Privacy: A Case Study | 142 |
| <i>Ikuto Takashima, Kotaro Yasuda, Yukiko Takeuchi, Masayoshi Aritsugi, Akihiro Shibayama, and Israel Mendonça</i> | |
| Unsupervised Segmentation of CNC Milling Sensor Data into Comparable Cutting Conditions | 149 |
| <i>Manuel Götz, Maximilian Rost, Dennis Wilkner, and Frank Schirmeier</i> | |
| Data Security and Privacy | |
| A Data-Driven Approach for Designing Wireless Signal Propagation Model ... | 159 |
| <i>Zhihui Wang, Wenbiao Xing, Yu Wang, and Yishan Liu</i> | |
| Identifying Personal Identifiable Information (PII) in Unstructured Text: A Comparative Study on Transformers | 174 |
| <i>Md Hasan Shahriar, Anne V. D. M. Kayem, David Reich, and Christoph Meinel</i> | |
| Database Indexing and Query Processing | |
| Unlocking the Power of Diversity in Index Tuning for Cluster Databases | 185 |
| <i>Haitian Hang, Xiu Tang, Bo Zhou, and Jianling Sun</i> | |
| SAFE: Sampling-Assisted Fast Learned Cardinality Estimation for Dynamic Spatial Data | 201 |
| <i>Yuchen Ji, Daichi Amagata, Yuya Sasaki, and Takahiro Hara</i> | |
| Lightweight Latches for B-Trees to Cope with High Contention | 217 |
| <i>Amir El-Shaikh, Bernhard Seeger, and Eljas Soisalon-Soininen</i> | |

| | |
|---|------------|
| <p>R-DiP: Re-ranking Based Diffusion Pre-computation for Image Retrieval</p> <p><i>Tatsuya Kato, Takahiro Komamizu, and Ichiro Ide</i></p> | <p>233</p> |
| <p>Extension of Parallel Primitives and Their Applications to Large-Scale Data Processing</p> <p><i>Masashi Nakano, Qiong Chang, and Jun Miyazaki</i></p> | <p>248</p> |
| <p>Specialized Applications and Case Studies</p> | |
| <p>Fast Concept Drift Detection Exploiting Product Quantization</p> <p><i>Taisei Takano and Hisashi Koga</i></p> | <p>257</p> |
| <p>A Novel Multi-scale Spatiotemporal Graph Neural Network for Epidemic Prediction</p> <p><i>Zenghui Xu, Mingzhang Li, Ting Yu, Linlin Hou, Peng Zhang, Rage Uday Kiran, Zhao Li, and Ji Zhang</i></p> | <p>272</p> |
| <p>LALO—A Virtual Data Lake Zone for Composing Tailor-Made Data Products on Demand</p> <p><i>Christoph Stach, Yunxuan Li, Laura Schuiki, and Bernhard Mitschang</i></p> | <p>288</p> |
| <p>Intermediate Hidden Layers for Legal Case Retrieval Representation</p> <p><i>Eya Hammami, Mohand Boughanem, Rim Faiz, and Taoufiq Dkaki</i></p> | <p>306</p> |
| <p>Geometric Features and GAT Neural Network for Protein Surface Classification</p> <p><i>Wissam Ferroudj, Noura Faci, and Hamamache Kheddouci</i></p> | <p>320</p> |
| <p>Author Index</p> | <p>327</p> |

Financial and Economic Data Analysis



CSPRD: A Financial Policy Retrieval Dataset for Chinese Stock Market

Jinyuan Wang¹, Zhong Wang^{2(✉)}, Zeyang Zhu², Jinhao Xie², Yong Yu²,
Yongjian Fei², Yue Huang², Dawei Cheng³, and Hai Zhao¹

¹ Shanghai Jiao Tong University, Shanghai, China
steve.wang@sjtu.edu.cn

² Shanghai Stock Exchange Technology Co., Ltd., Shanghai, China
zhongwang@sse.com.cn

³ Tongji University, Shanghai, China

Abstract. Recently, Large language models (LLMs) have demonstrated formidable capabilities, yet challenges persist in real-world applications, particularly in aspects of hallucination, misinformation and outdated knowledge. Retrieval-Augmented Generation (RAG) addresses these challenges by pre-retrieving pertinent information from external knowledge bases prior to utilizing LLMs for answering queries. Although RAG has been empirically validated to enhance response accuracy and reduce error rates, the paucity of domain-specific datasets hampers the development of proficient retrievers, therefore becoming a bottleneck in deploying RAG pipelines within professional fields. In this work, we propose a novel task termed stock policy retrieval and introduce the Chinese Stock Policy Retrieval Dataset (CSPRD), comprising 700+ prospectus excerpts annotated by seasoned experts, correlated with relevant articles from a collection of more than 10,000 entries in our amassed Chinese policy corpus. Our experiments with lexical, embedding, and fine-tuned bi-encoder models not only attests to the efficacy of our proposed CSPRD but also indicates considerable potential for enhancement. To capitalize on high quality encodings, we proposed CSPR-MQA, a retrieval-oriented pre-training paradigm that amalgamates various supervised natural language processing (NLP) tasks into an unsupervised framework for masked question-answering (MQA). Our CSPR-MQA model, after pre-training on 61GB Chinese corpus and fine-tuning on CSPRD, achieves the best performance on the CSPRD development set, with metrics including 57.9% MRR@10, 29.1% NDCG@10 and 39.3% Recall@10.

Keywords: Information Retrieval · Retrieval Augmented Generation · Decision Support Systems

1 Introduction

Recent years have witnessed Large language models (LLMs) such as the GPT series [1, 2] and Llama series [25], demonstrating exceptional linguistic pro-

This work was supported by the National Key Research and Development Program of China (2021YFC3340700).

iciency and knowledge understanding capabilities across various evaluation benchmarks, which surpass several human evaluation standards [11, 26].

However, these generative LLMs still face significant deficiencies in the authenticity and accuracy of content generation. Despite the substantial achievements of LLMs in capturing language structure and emulating human writing styles, they exhibit significant limitations, particularly in handling domain-specific or highly specialized queries [8, 14]. When the required information exceeds the scope of the models’ training data or necessitates the latest data, LLMs might provide inaccurate answers or misinformation. Such limitations present significant challenges in deploying generative artificial intelligence in real-world applications, especially in specialized fields such as finance, law, and healthcare [20].

Retrieval-Augmented Generation (RAG) has emerged as a promising solution by merging a pre-trained retriever with a pre-trained sequence-to-sequence model (generator), capturing knowledge in a more interpretable and modular fashion through end-to-end fine-tuning. A typical RAG pipeline comprises a retriever and a generator, often fulfilled by a LLM. However, the scarcity of specialized domain datasets restricts the development of proficient, specialized retrievers, thereby becoming a bottleneck in deploying effective and efficient RAG pipelines in specialized domains.

In this work, we propose a new retrieval task, namely stock policy retrieval, which requires a set of matched policy articles from a large corpus given a passage concerning the primary business and principal products in a company’s prospectus. To further build a proficient policy retriever, we proposed CSPR-MQA, an innovative retrieval-oriented pre-training paradigm that amalgamates various supervised natural language processing (NLP) tasks into an unsupervised framework for masked question-answering (MQA).

A qualified policy retrieval system should not only offer professional auxiliary services for regulatory agencies, but also provide investors with more thorough information for investment decisions. However, finding policy articles that match the given business description can be a rigorous task as there are two key challenges. (1) Different from general commonsense retrieval [21], stock policy retrieval needs to deal with two types of differently distributed languages [20]: complex yet plain language for prospectuses and concise yet fragmentary language for policies. Addressing such difference indirectly demands an almighty system that can not only focus on key information in complex scenario but translate concise expressions into natural language as well. (2) The prospectus passages include vague industry identification and specific product description of the company. However, a policy item match is not solely relied on the accordance of industrial category, but rather on the consistency of business products.

Therefore, a large-scale policy retrieval dataset with expert annotations is necessary to study the extent to which retrieval models can pair with the wisdom and decision-making of professional analysts in regulatory agencies. Admittedly, government policies and prospectuses of listed companies are publicly accessible. However, the review process by regulatory agencies is often a black box, and the matched policy articles of listed companies are not publicly available, which

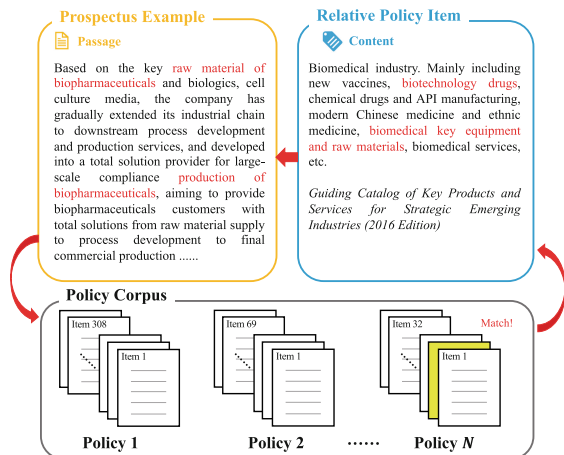


Fig. 1. Illustration of the policy retrieval task performed on the Chinese Stock Policy Retrieval Dataset (CSPRD), which consists of 700+ prospectus passages carefully labeled by experienced experts with references to relevant policy articles collected by the Shanghai Stock Exchange.

set a high barrier to collecting such datasets. To the best of our knowledge, we are the first work to introduce a policy retrieval dataset, namely the Chinese Stock Policy Retrieval Dataset (CSPRD), filling the blank of fact-driven retrieval dataset in financial and stock market. The main contributions of this work are threefold (Fig. 1):

- We introduce the Chinese Stock Policy Retrieval Dataset (CSPRD), comprising a Chinese policy corpus of 10,002 articles and 709 prospectus excerpts from 545 companies listed on Chinas Science and Technology Innovation Board (STAR Market). CSPRD is annotated by experienced experts from Shanghai Stock Exchange (SSE) and released bilingual in Chinese and English¹.
- We propose CSPR-MQA, an innovative retrieval-oriented pre-training paradigm with asymmetric dual-decoder structure and asymmetric masking strategy [19] to further improve the encoding ability of PLMs on our proposed CSPRD dataset.
- We establish strong baselines on the CSPRD dataset by benchmarking several state-of-the-art retrieval approaches, including lexical, embedding and fine-tuning models. Our proposed CSPR-MQA is the best among the baselines by achieving 57.9% MRR@10, 29.1% NDCG@10, 39.3% Recall@10 and 81.9% Precision@10, which shows the effectiveness of our proposed CSPR-MQA yet also suggests ample potential for improvement.

Our code and dataset is publicly available on GitHub².

¹ Translated by ChatGPT (gpt-3.5-turbo-16k-0613).

² https://github.com/noewangjy/csprd_dataset.

2 Related Work

2.1 Retrieval Augmented Generation

Within the realm of LLMs, RAG stands as a model enhancement paradigm [8, 9, 12, 17, 18], which involves integrating an information retrieval component with a text generation model, utilizing factual information as input to the generative large models to enhance the factuality and accuracy of their outputs. The essence of the RAG model lies in the amalgamation of traditional generative language models with information retrieval systems. Specifically, upon receiving an input (such as a question or prompt), it first employs a retrieval system to find relevant information from a large document collection. Subsequently, these documents are used as context to aid the generative model in producing more accurate and in-depth text. RAG processes the input and retrieves a set of documents from given sources (e.g., Wikipedia), which are then combined with the original input prompt as context and fed into the text generator to produce the final output.

The primary advantage of the RAG pipeline lies in its ability to integrate the creativity of generative models with the factuality based on retrieved information [8, 9, 18]. This combination offers enhanced factual accuracy, strengthened capabilities in specific domains and application flexibility in various scenarios. However, the main limitations of RAG technology are concentrated in two aspects: **(1). Proficient Retriever:** it has high demands on the performance of the retriever, which must be capable of retrieving a large volume of relevant knowledge to ensure a high recall rate, while also minimizing false positives to ensure accuracy [15, 28]; **(2). Domain-specialized Corpus:** most existing retrieval datasets focus on general knowledge retrieval, and common sense question-answering datasets are often used to benchmark models [16, 21]. However, research in areas like finance and economics is far from deep due to the lack of large volumes of high-quality datasets and professional annotations [20].

This work is devoted to addressing the above limitations by proposing enhancements to the RAG framework that focus on expanding the scope of domain-specialized corpora and improving the proficiency of retrievers in specialized domains.

2.2 Dense Retrievers

In general, a policy retriever is a function that takes a prospectus passage as input along with a corpus of policy articles and returns a small set of relevant policies.

Recent advances in pre-trained language models (PLMs) have sparked remarkable success in numerous NLP tasks [7]. Building upon PLM-based encoders, dense retrieval [15, 29] has been proven an effective paradigm for dense retrievers [19, 22], which aims to retrieve correlated passages of a given query from a massive corpus. However, most existing retrieval datasets substantially benchmark models on general commonsense retrieval [21], while specialized

domains such as finance and economics remain unexplored due to deficiency of large-scale high-quality datasets with expert annotations [20].

In general, a policy retriever is a function that takes a prospectus passage as input along with a corpus of policy articles and returns a small set of relevant policies. Lexical approaches have been the de facto standard for textual retrieval for their robustness and efficiency, such as BM25 [23] and TF-IDF. In recent years, dense retrieval methods [15, 29] have been proven an effective paradigm in open-domain question-answering, which are built upon PLMs-based encoders. Karpukhin *et al.* [15] proposed DPR, which employs a bi-encoder design with in-batch contrastive learning training. Xiong *et al.* [29] proposed ANCE, a learning mechanism that selects hard training negatives globally from the entire corpus. Furthermore, retrieval-oriented pre-training methods [19, 22] are proposed to generate better textual encoding for textual retrieval. Among them, RocketQA [22] proposes optimized training approaches to address discrepancy between training and inference. RetroMAE [19] adopt asymmetric masked encoder-decoder design and unbalanced masking ratios during pre-training.

2.3 Specialised Financial Datasets

Recently, more and more domain-specific datasets are introduced in the NLP community to enrich fact-driven datasets in financial tasks, such as financial sentiment analysis [6], numerical reasoning [3] and multilingual topic classification [13].

Some existing works focus on textual information published by enterprises. Daudert *et al.* [5] introduced CoFiF dataset, which contains 2655 french reports in span of 20 years, covering reference documents, annual, semestrial and trimestrial reports. The JOCo corpus [10] is composed of corporate annual and social responsibility reports of top-ranked international companies. In terms of policy corpus, Wilson *et al.* [27] created a corpus of 115 privacy policies with manual annotations for fine-grained data practices. However, there is few attention on the policy compliance of enterprise prospectus. To fill this blank, this work is devoted to introducing a Chinese policy retrieval datasets with expert annotations in stock market.

3 The Policy Retrieval Dataset for Stock Market in China

In this section, we detail the process of creating the CSPRD dataset, which includes the following five stages.

3.1 Data Collection

In June 2019, the STAR Market was officially established by SSE, where listed companies are conform to certain incentive policy articles of Chinese government due to the requirement of SSE. From the public information on the official

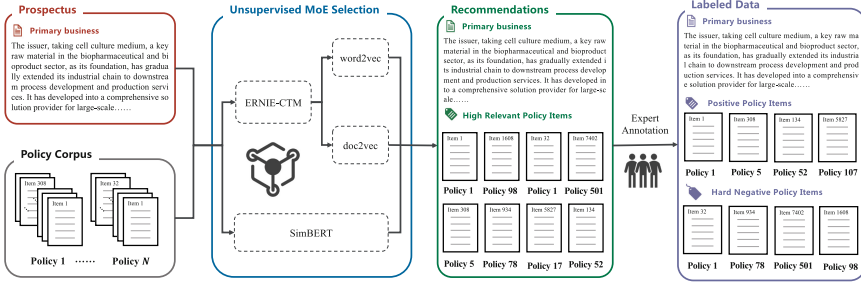


Fig. 2. Overview of the annotation process. After data processing, the collected prospectus passages and policy articles are fed to a mixture-of-experts (MoE) selection system composed of unsupervised models. The Top-20 ranked policy articles for each prospectus passage are selected as recommendation for the human annotation process.

website³ of the STAR Market, we collect the prospectuses of 767 listed companies, as well as 400+ incentive policy documents compiled by SSE (as of August, 2022). The policy documents are classified into 7 categories by SSE, including *New Generation of Science and Technology*, *High-end Equipment*, *New Materials*, *New Energy*, *Environment Protection*, *Biomedicine* and *General* (Fig. 2).

3.2 Data Processing

For prospectuses, we solely extract the textual information concerning key products and services. We conduct semantic-based keyword matching and positioning through file metadata, and specifically extract the text content under the corresponding chapter title as contextual information based on reference keywords such as *main products* and *primary business*. To improve the quality and accuracy of chapter positioning, we extract and filter the textual chunks after truncating the PDF files by matched chapter titles.

As for policy documents, we use regular expressions to match the policy names and split them into policy articles by title. We manually set block words to filter out political-relevant and financial-irrelevant articles.

3.3 Unsupervised MoE Selection

In the annotation step, each prospectus excerpt is paired with each policy article, which is in total 7 million pairs to be scored. To significantly reduce cost of human resources, we deploy a decision support system with a mixture of experts (MoEs) to directly score the textual similarity of each pair of prospectus passage and policy content.

Our MoE selection system is consisted of unsupervised models trained on both the policy corpus and prospectus excerpts. We first adopt ERNIE-CTM

³ <https://kcb.sse.com.cn/>.

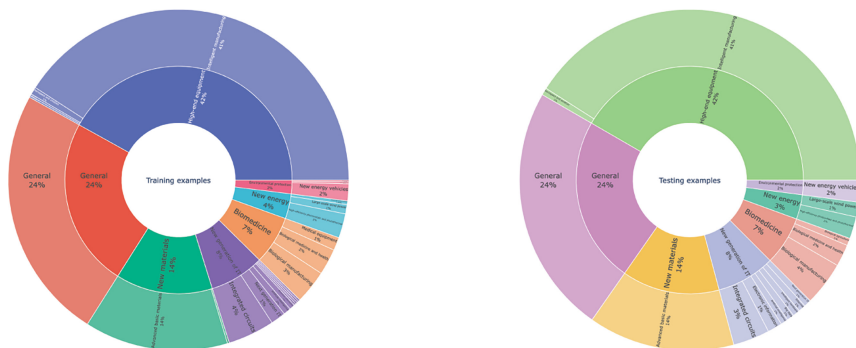


Fig. 3. Category distribution of the examples in CSPRD.

[32] to recognize named entities in the excerpts and only keeps the named entities with manually selected tags. Then the named entities are used to train a `word2vec` model and a `doc2vec` model, which are used to score the textual similarity. In addition to that, we employ a pre-trained SimBERT⁴ [24] to directly score the similarity. At the rear of the system, the final score for each text pair is the weighted sum of the scores given by `word2vec` (10%), `doc2vec` (20%) and SimBERT (70%). As recommendation, we choose the 20 top-ranking policy articles for each prospectus excerpt (Fig. 3).

3.4 Expert Annotation

The CSPRD is annotated by 5 experienced SSE experts, who have systematically studied the manual *QAs on the Review of Stock Issuance and Listing on the SSE STAR Market*⁵. During annotation, each expert is required to focus on the primary products, main business and specific core technologies in the prospectus excerpt and judge whether they are compliant with the specific industry and target applications in the recommended policy articles. After cautious reading and thorough judgement, the experts should choose one of the ternary labels ([Yes], [No], [Uncertain]). Each policy pair is independently labeled by one expert and policy pairs labeled with [Uncertain] will be re-collected and re-labeled through group discussion of all experts. After expert annotation, the policy articles labeled as [Yes] are positive articles, while those labeled as [No] are referred as hard negative ones for contrastive learning.

3.5 Dataset Release

CSPRD contains a Chinese policy corpus of 10,002 articles and 709 prospectus examples from 545 companies listed on the STAR Market in China. CSPRD

⁴ <https://github.com/PaddlePaddle/PaddleNLP>.

⁵ <http://www.sse.com.cn/lawandrules/sserules/tib/review/c/4729640.shtml>.

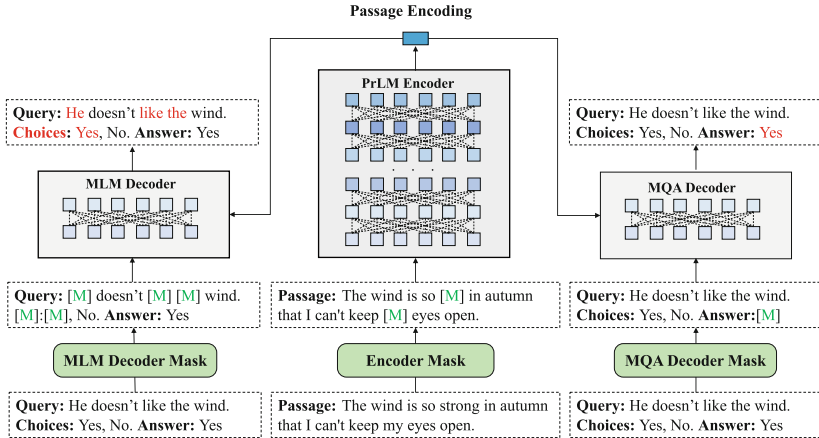


Fig. 4. Illustration of our proposed CSPR-MQA paradigm. The encoder adopts a PLM whose input is moderately masked. The dual-decoder consists of 2 one-layer transformer whose reference is the passage encoding from the encoder. The MLM decoder apply an aggressive masking ratio, while the MQA only masks all the answer tokens.

is bilingual in Chinese and English: the origin language of CSPRD is simplified Chinese, and the English version is translated by ChatGPT⁶ with direct prompting. The English version is for research purpose only, the translation quality has no assurance from authors. We select 80% data of each category as the train set, while the remaining 144 examples as the dev set (Fig. 4).

4 CSPR-MQA Pre-training

In this section, we introduce an aggressive method of masked language modeling (MLM) namely masked question answering (MQA), which is designed to be a rigorous reconstruction task to further enhance encoding quality. Inspired by [19], we propose CSPR-MQA, a novel retrieval oriented pre-training paradigm with asymmetric dual-decoder structure and asymmetric masking strategy. CSPR-MQA adopts a full-scale PLM as its encoder and two identical one-layer transformer as its dual-decoder, namely MLM decoder and MQA decoder.

4.1 Data Preprocessing

MQA aims to unify various forms of supervised NLP tasks⁷ into an unsupervised form. We represent each sample as a quadruple $[P, Q, C, A]$, where P for the passage, Q for the query (automatically generated), C for the choices (possibly empty), and A for the answer. We design customized questions for specific supervised tasks and fixed questions for unsupervised corpus (Table 1).

⁶ <https://openai.com/blog/chatgpt>.

⁷ <https://github.com/ningshixian/NLP-zoo>.

Table 1. Overview of preprocessed data for masked question-answering (MQA). Each sample is formulated as a unified quadruple $[P, Q, C, A]$, where P for the passage, Q for the query (automatically generated), C for the choices (possibly empty), and A for the answer.

| Task | Supervision | Passage | Query | Choice | Answer |
|---------------|-------------|----------|------------|------------|-------------|
| CLUE-C3 | Yes | Joint | Original | Original | Original |
| CLUE-iFLYTEK | Yes | Original | Customized | – | Transferred |
| CLUE-tnews | Yes | Joint | Customized | – | Transferred |
| CLUE-CMRC2018 | Yes | Original | Original | – | Original |
| CLUE-AFQMC | Yes | Joint | Customized | Yes or No | Transferred |
| CLUE-CMNLI | Yes | Joint | Customized | Customized | Transferred |
| CLUE-CSL | Yes | Joint | Customized | – | Joint |
| CLUE-DRCD | Yes | Original | Original | – | Original |
| CLUE-CHID | Yes | Modified | Customized | – | Transferred |
| CLUE-WSC2020 | Yes | Original | Customized | Yes or No | Transferred |
| Dureader | Yes | Selected | Customized | – | Original |
| News2016 | No | Original | Customized | – | Joint |
| Wikipedia-zh | No | Modified | Customized | – | Customized |

4.2 Encoding

Given the passage P , we tokenize it into sequence X_P with whole word masking (WWM), where a small fraction (15%-30%) of its tokens will be replaced with [MASK] for MLM tasks. Then we apply a BERT like PLM as encoder and we select the [CLS] token in the last layer as the passage encoding H_P .

4.3 Decoding

MLM Decoding. We first concatenate the triplet $[Q, C, A]$ with prompts and then tokenize it into sequence X_{QCA} with WWM, where half of the tokens will be replaced with [MASK] token. We adopt a one-layer Transformer with enhanced decoding [19] as MLM Decoder, which is required to restore the aggressively masked X_{QCA} by referring the passage encoding H_P .

MQA Decoding. We apply almost the same concatenation, tokenization and masking as MLM Decoder, except that only the answer tokens are completely masked. We adopt another same single-layer Transformer decoder to restore the answer tokens X_A to the question with the passage encoding H_P as reference. Such long consecutive token prediction is quite challenging, therefore, it urges the system on higher quality encoding.

5 Experiments

In this section, we report the performance of lexical models, embedding models and fine-tuned PLMs on CSPRD dataset as baselines for future works. We

evaluate the retrieval performance with four commonly used metrics for information retrieval: mean reciprocal rank (MRR@10), normalized discounted cumulative gain (NDCG@10), recall (R@10) and precision (P@10).

5.1 Models

Given an example pair (P, A) and a policy corpus \mathcal{C} , where P is the prospectus passage, A is the policy article. We define the relevance score $r(P, A)$ for each method below.

Lexical Methods: For lexical methods, the relevance score is defined as the sum over the passage terms:

$$r(P, A) = \sum_{t \in P} w(t, A) \quad (1)$$

We calculate the weight with TF-IDF and BM25 [23] approaches respectively.

Embedding Methods: For embedding methods, the relevance score is defined as the cosine similarity of the embeddings:

$$r(P, A) = \text{CosineSimilarity}(E(P), E(A)) \quad (2)$$

where $E(\cdot)$ is the embedding model.

We use the texts in C SPRD dataset to fit a `word2vec` (W2V-C SPRD) and a `doc2vec` (D2V-C SPRD) models and test their performance on the dev set. In addition to that, we also benchmark an open-sourced W2V embedding model (W2V-Finance⁸) trained on finance texts.

Fine-Tuning Methods: The relevance score of embedding methods is defined as the softmax score of the inner product of the encoding matrices:

$$r(P, A) = \text{Softmax}_{A \in \mathcal{C}}(E(P) \cdot E(A)^T) \quad (3)$$

where $E(\cdot)$ is the encoding function of the fine-tuned models.

For embedding methods, we implement a bi-encoder framework with different PLMs in size of BERT_{base} [7] as encoder. Our models are fine-tuned on the train set of C SPRD with in-batch negative contrastive learning proposed in DPR [15].

Since the open-sourced RetroMAE [19] is pre-trained on English corpus, we pre-train a RetroMAE model from scratch on ~ 61 GB publicly collected Chinese corpus and then fine-tune it on the train set of C SPRD dataset. We adopt the pre-trained Chinese BERT [4] as encoder, which was pre-trained with whole word masking (WWM) [4] on extra ~ 5.4 B words of Chinese corpus. During our pre-training, we respect the WWM strategy to keep consistency with the previous pre-training. In the subsequent pre-training task, the model is pre-trained for 5 epochs with learning rate of $1e^{-4}$ and weight decay of 0.01. During fine-tuning, each model is fine-tuned for 10 epochs with learning rate of $2e^{-5}$.

⁸ <https://github.com/Embedding/Chinese-Word-Vectors>.

5.2 Evaluation Metrics

This paper employs four commonly used metrics for assessing the performance of information retrieval and recommendation systems: Mean Reciprocal Rank at 10 (MRR@10), Normalized Discounted Cumulative Gain at 10 (NDCG@10), Recall at 10 (R@10), and Precision at 10 (P@10).

MRR@n (Mean Reciprocal Rank at n): MRR@n calculates the average of the reciprocal ranks of the first relevant document for each query in a query set Q . Here, rank_i represents the rank of the first relevant document for the i th query. If no relevant document appears within the top n results, then the contribution of that query is zero. MRR prioritizes documents ranked higher. The formula for calculation is:

$$\text{MRR@n} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i} \quad (4)$$

NDCG@n (Normalized Discounted Cumulative Gain at n): NDCG@n is a metric that considers both the relevance of documents and their rank positions. It penalizes poor ordering by diminishing the weight of documents ranked lower. IDCG@n represents the ideal DCG@n, used to normalize the results so that scores range between 0 and 1. The calculation formula is:

$$\text{NDCG@n} = \frac{\text{DCG@n}}{\text{IDCG@n}} \quad (5)$$

where:

$$\text{DCG@n} = \sum_{i=1}^n \frac{2^{\text{rel}_i} - 1}{\log_2(i + 1)} \quad (6)$$

Recall@n: Recall@n measures the proportion of relevant documents retrieved within the top n results against the total number of relevant documents. It focuses on retrieving as many relevant documents as possible. The formula for calculation is as follows:

$$\text{Recall@n} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|} \quad (7)$$

Precision@n: Precision@n measures the proportion of relevant documents within the top n retrieved results against the total number of retrieved documents. It focuses on the accuracy of the retrieval results. The formula for calculation is as follows:

$$\text{Precision@n} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|} \quad (8)$$

5.3 Results and Analysis

Our experiment results are shown in Table 2. As de facto standard methods, lexical methods TF-IDF and BM25 show poor performance on our CSPRD dataset, suggesting the challenge of our proposed policy retrieval task.

We discover that there is a positive correlation between policy relevance and textual similarity of policy articles and prospectus passages. However, models that exhibit good performance in textual similarity, without further fine-tuning, still fail to achieve satisfactory results than fine-tuned models.

Traditional methods perform rather poorly, indicating that attempting to address this task purely from the statistics of term frequency is quite challenging. Large language models (LLMs) are decoder-only models, while such task requires strong encoding capability, therefore, LLMs are not suitable for this task. For this particular task, fine-tuning smaller models is still necessary and efficient.

Table 2. Retrieval benchmark of several approaches on CSPRD dev set. We pre-trained RetroMAE [19] from scratch on ~61GB Chinese corpus with Chinese BERT [4] encoder. The models with † are fine-tuned with DPR [15] framework.

| Model | MRR@10 | NDCG@10 | Recall@10 | Precision@10 |
|------------------|-------------|-------------|-------------|--------------|
| TF-IDF | 3.2 | 1.6 | 2.2 | 1.6 |
| BM25 [23] | 22.3 | 13.1 | 14.3 | 13.1 |
| W2V-CSPRD | 9.9 | 11.2 | 5.3 | 18.6 |
| D2V-CSPRD | 10.3 | 5.2 | 6.0 | 5.2 |
| W2V-Finance | 19.8 | 9.9 | 10.4 | 9.9 |
| BERT† [4, 7] | 53.6 | 26.9 | 35.8 | 79.2 |
| MacBERT† [4] | 50.4 | 25.4 | 34.7 | 79.2 |
| Mengzi† [31] | 52.1 | 26.2 | 35.6 | 82.6 |
| RetroMAE† [19] | 54.8 | 27.1 | 35.8 | 79.9 |
| CoSENT† [30] | 56.1 | 28.5 | 37.5 | 80.6 |
| CSPR-MQA† (Ours) | 57.9 | 29.1 | 39.3 | 81.9 |

6 Limitations

As far as we are aware of, this work has the following 2 limitations:

6.1 Compromise Between Labor Costs and Decision Comprehensiveness

This study employs three unsupervised methods to calculate the weighted scores of text similarity for recommendation ratings, while human experts only evaluate whether the top 20 suggested policy entries match the paragraphs of the prospectus. The annotation in this paper is based on the assumption that the

relevance between a given prospectus paragraph and a given policy entry is predicated on their text similarity. However, this assumption is made to significantly reduce labor costs and minimize interference with human annotators rather than based on statistical conclusions. This is because the research engaged experts from the Shanghai Stock Exchange as full-time annotators, which incurs higher labor costs compared to crowd-sourced annotation methods. Therefore, for a given prospectus paragraph, there might be relevant policy entries that are not marked. However, in real-world scenarios, such compromise on comprehensiveness is inevitable, as regulatory auditors are more likely to refer to textually similar entries rather than recall all policy entries for judgment.

6.2 Limited Experiments on Pre-Trained Language Models

This study opted for the Chinese BERT series models [7] as the default pre-trained language model encoders, and most experiments were conducted under default settings. Despite the existence of better or larger Chinese pre-trained language models, they have not undergone Whole Word Masking (WWM) pre-training. This paper posits that WWM pre-training aligns with the increasing trend from subword masking to clause masking, and even to whole sentence masking, increasing the continuous masked token lengths. Therefore, pre-trained language models that have undergone WWM are capable of bridging the gap between short and long sequences of continuous masked tokens. It is noteworthy that subsequent derivatives of BERT might yield better performance; however, due to limited computational resources, this paper only selected the Chinese BERT as the base model for a comprehensive set of tests.

7 Conclusion

In this paper, we introduce the Chinese Stock Policy Retrieval Dataset (CSPRD), a compilation of over 700 prospectus passages accompanied by pertinent policy articles meticulously annotated by experts from the Shanghai Stock Exchange. We assessed numerous information retrieval baselines, demonstrating the utility and promise of CSPRD dataset. In order to improve the encoding quality, we proposed a retrieval oriented pre-training paradigm, CSPR-MQA, which achieves significant better scores on CSPRD after pre-training by our designed masked question-answering (MQA) task. Our work bridges a notable gap in the realm of financial datasets for NLP and paves way for future study on policy retrieval task.

References

1. Achiam, J., et al.: GPT-4 technical report. arXiv preprint [arXiv:2303.08774](https://arxiv.org/abs/2303.08774) (2023)
2. Brown, T., et al.: Language models are few-shot learners. *Adv. Neural. Inf. Process. Syst.* **33**, 1877–1901 (2020)

3. Chen, Z., et al.: FinQA: a dataset of numerical reasoning over financial data. arXiv preprint [arXiv:2109.00122](https://arxiv.org/abs/2109.00122) (2021)
4. Cui, Y., Che, W., Liu, T., Qin, B., Wang, S., Hu, G.: Revisiting pre-trained models for Chinese natural language processing. In: Findings of the Association for Computational Linguistics: EMNLP 2020, pp. 657–668. Association for Computational Linguistics, Online (2020). <https://doi.org/10.18653/v1/2020.findings-emnlp.58>, <https://aclanthology.org/2020.findings-emnlp.58>
5. Daudert, T., Ahmadi, S.: CoFiF: a corpus of financial reports in French language. In: Proceedings of the First Workshop on Financial Technology and Natural Language Processing, pp. 21–26. Macao, China (2019), <https://aclanthology.org/W19-5504>
6. Daudert, T., Buitelaar, P., Negi, S.: Leveraging news sentiment to improve microblog sentiment classification in the financial domain. In: Proceedings of the First Workshop on Economics and Natural Language Processing, pp. 49–54. Association for Computational Linguistics, Melbourne (2018). <https://doi.org/10.18653/v1/W18-3107>, <https://aclanthology.org/W18-3107>
7. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. arXiv preprint [arXiv:1810.04805](https://arxiv.org/abs/1810.04805) (2018)
8. Gao, Y., et al.: Retrieval-augmented generation for large language models: a survey. arXiv preprint [arXiv:2312.10997](https://arxiv.org/abs/2312.10997) (2023)
9. Guu, K., Lee, K., Tung, Z., Pasupat, P., Chang, M.: Retrieval augmented language model pre-training. In: International Conference on Machine Learning, pp. 3929–3938. PMLR (2020)
10. Händschke, S.G., et al.: A corpus of corporate annual and social responsibility reports: 280 million tokens of balanced organizational writing. In: Proceedings of the First Workshop on Economics and Natural Language Processing, pp. 20–31. Association for Computational Linguistics, Melbourne (2018). <https://doi.org/10.18653/v1/W18-3103>, <https://aclanthology.org/W18-3103>
11. Hendrycks, D., et al.: Measuring massive multitask language understanding (2021)
12. Jiang, Z., et al.: Active retrieval augmented generation. arXiv preprint [arXiv:2305.06983](https://arxiv.org/abs/2305.06983) (2023)
13. Jørgensen, R., Brandt, O., Hartmann, M., Dai, X., Igel, C., Elliott, D.: MultiFin: a dataset for multilingual financial NLP. In: Findings of the Association for Computational Linguistics: EACL 2023, pp. 894–909. Association for Computational Linguistics, Dubrovnik (2023). <https://aclanthology.org/2023.findings-eacl.66>
14. Kandpal, N., Deng, H., Roberts, A., Wallace, E., Raffel, C.: Large language models struggle to learn long-tail knowledge (2023)
15. Karpukhin, V., et al.: Dense passage retrieval for open-domain question answering. arXiv preprint [arXiv:2004.04906](https://arxiv.org/abs/2004.04906) (2020)
16. Kwiatkowski, T., et al.: Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguist.* **7**, 453–466 (2019)
17. Lewis, P., et al.: Retrieval-augmented generation for knowledge-intensive NLP tasks. *Adv. Neural Inf. Process. Syst.* **33**, 9459–9474 (2020)
18. Liu, S., Chen, Y., Xie, X., Siow, J., Liu, Y.: Retrieval-augmented generation for code summarization via hybrid GNN. arXiv preprint [arXiv:2006.05405](https://arxiv.org/abs/2006.05405) (2020)
19. Liu, Z., Shao, Y.: RetroMAE: pre-training retrieval-oriented transformers via masked auto-encoder. arXiv preprint [arXiv:2205.12035](https://arxiv.org/abs/2205.12035) (2022)
20. Louis, A., Spanakis, G.: A statutory article retrieval dataset in french. arXiv preprint [arXiv:2108.11792](https://arxiv.org/abs/2108.11792) (2021)

21. Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., Deng, L.: MS marco: a human generated machine reading comprehension dataset. In: CoCo@ NIPs (2016)
22. Qu, Y., et al.: RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. arXiv preprint [arXiv:2010.08191](https://arxiv.org/abs/2010.08191) (2020)
23. Robertson, S., Walker, S., Jones, S., Hancock-Beaulieu, M.M., Gatford, M.: Okapi at TREC-3. In: Overview of the Third Text REtrieval Conference (TREC-3), pp. 109–126. NIST, Gaithersburg (1995). <https://www.microsoft.com/en-us/research/publication/okapi-at-trec-3/>
24. Su, J.: SimBERT: integrating retrieval and generation into BERT. Technical report (2020). <https://github.com/ZhuiyiTechnology/simbert>
25. Touvron, H., et al.: LLaMA: open and efficient foundation language models. arXiv preprint [arXiv:2302.13971](https://arxiv.org/abs/2302.13971) (2023)
26. Wang, A., et al.: SuperGLUE: a stickier benchmark for general-purpose language understanding systems (2020)
27. Wilson, S., et al.: The creation and analysis of a website privacy policy corpus. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1330–1340. Association for Computational Linguistics, Berlin (2016). <https://doi.org/10.18653/v1/P16-1126>, <https://aclanthology.org/P16-1126>
28. Wu, B., Zhang, Z., Wang, J., Zhao, H.: Sentence-aware contrastive learning for open-domain passage retrieval. In: Muresan, S., Nakov, P., Villavicencio, A. (eds.) Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1062–1074. Association for Computational Linguistics, Dublin (2022)
29. Xiong, L., et al.: Approximate nearest neighbor negative contrastive learning for dense text retrieval. arXiv preprint [arXiv:2007.00808](https://arxiv.org/abs/2007.00808) (2020)
30. Xu, M.: Text2vec: text to vector toolkit (2023). <https://github.com/shibing624/text2vec>
31. Zhang, Z., et al.: Mengzi: towards lightweight yet ingenious pre-trained models for Chinese. arXiv preprint [arXiv:2110.06696](https://arxiv.org/abs/2110.06696) (2021)
32. Zhao, M., Qin, H., Zhang, G., Lyu, Y., Zhu, Y.: Termtree and knowledge annotation framework for Chinese language understanding. Technical report TR:2020-KG-TermTree, Baidu, Inc. (2020)



Leveraging Heterogeneous Text Data for Reinforcement Learning-Based Stock Trading Strategies

Keishi Fukuda^(✉) and Qiang Ma^(iD)

Kyoto Institute of Technology, Matasugasaki, Sakyo-ku, Kyoto 6068585, Japan
kfukuda@soc.is.kit.ac.jp, qiang@kit.ac.jp

Abstract. The number of individuals investing in stocks has increased due to the need for retirement asset-building and government recommendations. However, many of these investors are novices, making adequate stock trading support increasingly crucial. Existing systems for stock trading based on reinforcement learning primarily react to SNS posts or news that impact stock prices in short-term failing to leverage information that impacts stock prices in the medium- to long-term, such as earnings reports. This study proposes a reinforcement learning method for stock trading support that integrates texts affecting stock prices in the medium- to long-term, alongside texts impacting prices in the short-term. Our method updates the network that extracts features from these two types of texts, thereby acquiring strategies to assist stock trading. When applied to learning and testing stock trading scenarios, the proposed method demonstrates a higher return rate than existing methods and index investing.

Keywords: Stock trading · Heterogeneous Text Data · Investment Informatics · Reinforcement Learning

1 Introduction

The landscape of stock investment is witnessing a significant surge, driven by the growing necessity for retirement asset accumulation and governmental endorsements. Data from the TSE “Stock Distribution Survey” evidences a continuous rise in the number of individual shareholders in Japan, peaking at 69.82 million in 2022 [6]. Amidst this growth, bolstered by the Japanese government’s expansion of tax-free investment limits for 2024, most individual investors need essential investment knowledge, posing a challenge to efficient market participation. Surveys by the Japan Securities Dealers Association and The Investment Trusts Association of Japan in 2023 and 2020 reveal a stark gap in financial literacy and investment education among potential investors [7, 26].

Various stock trading support systems have been studied in response to the critical need for supporting novice and seasoned investors. Algorithmic trading, portfolio management, and order execution are some areas of stock trading

This work was partly supported by JSPS KAKENHI (23H03404 and 23K28094).

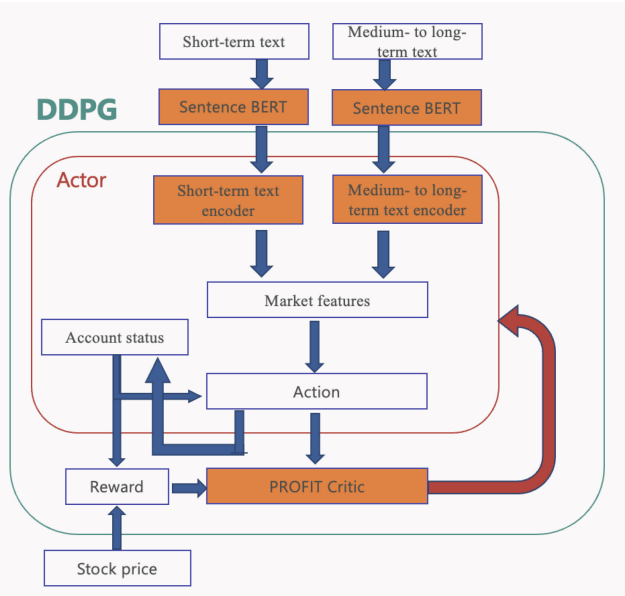


Fig. 1. Overview of Propose Method

support [14]. In recent years, research and development of stock trading support systems using reinforcement learning has been particularly active [24, 25]. However, existing research has yet to fully use information with medium- to long-term effects on stock prices, such as earnings reports. As a result, conventional systems have not been able to simultaneously consider events with short-term effects on stock prices and those with medium- to long-term effects on stock prices. For instance, Ramit et al. [24] only targets tweets and news, so it cannot fully consider medium- to long-term effects.

In this paper, we propose a novel reinforcement learning approach for stock trading that leverages both short-term (such as news and tweets) and medium- to long-term textual information (such as earnings reports) to inform trading decisions. Contrary to existing models that predominantly focus on immediate market reactions to social media posts and news, our method comprehensively encompasses events with enduring impacts on stock prices. Figure 1 shows an overview of the proposed method. The short-term and medium- to long-term texts are encoded and used as features that constitute the environment in DDPG [13] to learn strategies for stock trading. This dual-text strategy is encoded within a DDPG framework, offering a sophisticated tool for learning and executing stock trading strategies that account for a broader spectrum of market influencers.

The remainder of this paper is organized as follows: Sect. 2 introduces related work. Section 3 explains previous studies’ terminology, problem definition, and

terms. Section 4 introduces the proposed method. Section 5 presents experimental results and discusses them. Section 6 concludes this paper.

2 Related Work

Stock trading support systems are characterized by a rich tapestry of methodologies, ranging from traditional analyses to cutting-edge reinforcement learning techniques.

Kirihata et al. [8, 9] propose a method for detecting trends to estimate financial market conditions. Patel et al. [19] propose a method for predicting stock prices using supervised learning focusing on stock price volatility. Among the pioneering approaches in the realm of stock trading support systems, the heterotopic model proposed by Baba et al. [23] stands out for its innovative use of mixed data types. By integrating numerical data from stock prices with textual information from news articles, their model delves into the intricate relationship between firms' activities and market perceptions to forecast stock price movements.

The contributions of Takeda et al. [5], Lee et al. [11, 12], and Liu et al. [15, 16] pivot towards social trading services, emphasizing the importance of trader expertise and behavior in influencing trading strategies. Takeda et al. use a matrix factorization model to recommend that traders showcase an innovative application of machine learning to enhance social trading platforms. Similarly, Lee et al. and Liu et al. focus on identifying expert traders through portfolio theory, integrating traditional financial principles with contemporary computational techniques, offering a novel lens through which trader performance can be assessed within social trading environments.

Sakaji et al. [22] focus on leveraging syntactic patterns in economic newspaper articles to extract causal relationships using machine learning. This approach underscores the potential of natural language processing (NLP) in uncovering market insights from unstructured text data. Sakai et al. [21] extend the application of text analysis to financial statements, aiming to extract business performance indicators. Their work exemplifies how textual analysis can be applied to formal financial documents to predict company and market performance.

Filos et al. [4] and Cong et al. [17] both utilize reinforcement learning for managing stock portfolios, incorporating market trends and stock price fluctuations. These studies showcase reinforcement learning's capacity to adapt to and capitalize on complex market dynamics. Ramit et al. [24] and Gupta et al. [10] further the application of reinforcement learning by analyzing market trends through social media inputs like tweets and news. Their work demonstrates the relevance of real-time public sentiment and news in shaping market strategies. However, these methods use a single type of textual data as input and is insufficient to take into account diverse influences on the stock market.

This study proposes a novel reinforcement learning framework that harmonizes news and social media immediacy with the depth of earnings reports and financial disclosures. By doing so, we aim to offer a more holistic view of market

dynamics, enabling investors to make more informed decisions grounded in a fuller spectrum of market influencers.

3 Preliminaries

3.1 Terms

Short-term text Text that is expected to have a short-term impact on stock prices, such as posts and news.

Medium- to long-term text Text that is expected to have medium- to long-term effects on stock prices, such as a company’s earnings reports or economic indicators.

Market-information observation O_m Refers to Short-term text and Medium- to long-term text in the reinforcement learning context.

Stock features p The features of stocks obtained from the Short-term text and Medium- to long-term text.

Short-term stock features p'_s Features of stocks obtained from the Short-term text.

Medium- to long-term stock features p'_l Features of stocks obtained from the Medium- to long-term text.

Action Actions in the stock training include buying, selling, and holding.

Commission A generic term for trading and execution commission fee incurred when buying or selling.

Time step τ A time step (one day in our study) is the time unit for a trading model’s actions and updates.

Trading-account observation O_τ The balance in the trading account and the number of positions of each stock at time step τ .

Initial assets The assets at the time of the start of a stock trading. Initial assets consist of the balance in the trading account (cash).

Final assets The assets at the end of the trading. Final assets consist of the balance in the trading account (cash) and the total stock holdings at that time.

3.2 Problem Definition

The inputs to our trading model comprise a mixture of short-term texts, including tweets and news articles, and medium- to long-term texts, such as earnings reports, alongside stock prices. At each time step, the model evaluates these inputs-balancing the immediate insights from short-term texts and the strategic implications of medium- to long-term texts with the current stock prices-to determine the most advantageous trading actions. The output, a sophisticated stock trading support strategy, evolves continuously through reinforcement learning techniques. This dynamic adaptation ensures that the strategy remains optimized for both current market conditions and future expectations.

In this study, stock trading is formulated as a reinforcement learning problem based on the method of Ramit et al. [24]. State, action, and reward are defined as follows.

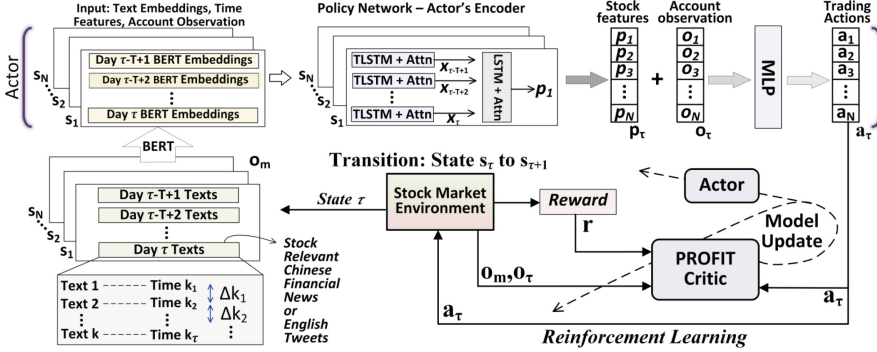


Fig. 2. Ramit method (adapted from [24])

State. At time τ , state s_τ consists of trading-account observation O_τ and market-information observation O_m . The trading-account observation O_τ is composed of the account balance and the number of holdings of each stock at time step τ . Market-information observation O_m consists of the short-term texts related to each stock announced during T days backward from time step τ and the N most recent medium- to long-term texts at time step τ .

Action. The agent takes one of three actions for each stock at time step τ : buy, sell, or hold.

Reward. Rewards are represented as the change in value when an action is taken in state s_τ and a new state $s_{\tau+1}$ is reached. Let r be the reward, a_τ the action at time step τ , b_τ the account balance, p_τ the stock price vector of a stock, h_τ the number of stocks held, and c the transaction fee, and define the reward $r(s_\tau, a_\tau, s_{\tau+1})$ as follows.

$$r(s_\tau, a_\tau, s_{\tau+1}) = (b_{\tau+1} + p_{\tau+1}^T h_{\tau+1}) - (b_\tau + p_\tau^T h_\tau) - c_\tau \quad (1)$$

3.3 Base Model

Here, we explain the quantification approach of Ramit et al. [24], which serves as the base model of our method. Hereafter, we will refer to this approach as the Ramit method.

Model. Figure 2 shows the trading model proposed in the Ramit method. Initially, tweets (i.e., posts) and other relevant texts undergo tokenization before being fed into a BERT-based encoder [3]. This encoder aggregates the final hidden layer outputs across all tokens to generate an encoded embedded representation. The significance of the timing and irregularity of tweets, as observed by O’Hara et al. [18], is addressed by employing a Time-aware Long Short-Term

Memory (TLSTM) model, as proposed by Baytas et al. [2], to capture the temporal irregularities inherent in tweets. Acknowledging the insight by Barber et al. [1] that not all tweets exert equal influence on market movements, Ramit method incorporates an attention layer to highlight texts with a more pronounced effect on stock prices. This refined output is then processed again through the TLSTM and attention layer to derive a market feature vector p_τ , representing the aggregated impact of tweets over the past τ days. The market feature vector p_τ is concatenated with the trading-account observation vector O_τ to compose the comprehensive state vector $s_\tau = [O_\tau, p_\tau]$. This vector s_τ serves as the input to a forward-propagating neural network, which employs a tanh activation function, and determines the action a_τ for each stock at the time step τ .

Training. Ramint method employs the Deep Deterministic Policy Gradient (DDPG) algorithm, as elucidated by Lillicrap et al. [13], to facilitate the training. The DDPG framework is adept at handling continuous action spaces, making it particularly suitable for stock trading models. In implementing the Ramit method, Ramit et al. adopt a distinct separation between action execution and performance evaluation, encapsulated by the Actor and PROFIT Critic components, respectively. The Actor is responsible for determining actions a_τ in any given state s_τ , which subsequently leads to a reward r_τ and transitions the model to the next state $s_{\tau+1}$. Meanwhile, the PROFIT Critic evaluates these actions by outputting a scalar value $Q(s_\tau, a_\tau)$, representing the expected return from state s_τ when action a_τ is taken. Transition experiences, denoted as tuples $(s_\tau, a_\tau, s_{\tau+1}, r_\tau)$, are accumulated in a buffer D . This repository facilitates experience replay, a crucial aspect of stabilizing the learning process. A mini-batch B consisting of N transitions is randomly sampled from D for model updates. This selection enables the efficient training of the network by approximating the gradient of the expected return with respect to the Actor’s parameters and updating the critic’s evaluation based on the temporal difference error. For each batch B , DDPG minimizes the following loss L with respect to ϕ and updates the PROFIT Critic as follows.

$$y_\tau = r_\tau + \gamma Q^{\phi'}(s_{\tau+1}, \mu^{\theta'}(s_{\tau+1})) \quad (2)$$

$$L = E[(y_\tau - Q^\phi(s_\tau, a_\tau))^2] \quad (3)$$

y_τ is the updated Q value. γ is the discount factor. θ and θ' , ϕ and ϕ' are the two copy parameters of the policy μ and the value function Q , respectively. The actor is updated using the policy gradient $\nabla_\theta J$ via backpropagation through time as follows.

$$\nabla_\theta J = E[\nabla_a Q^\phi(s_\tau, \mu^\theta(s_\tau)) \nabla_\theta \mu^\theta(s_\tau)] \quad (4)$$

4 Proposed Method

In this study, we expand the conventional scope of input data for stock trading models by incorporating tweets and news that exert short-term influences on

stock prices and earnings reports that have medium- to long-term effects. This approach recognizes the multifaceted nature of market dynamics, where immediate reactions to news events and the gradual assimilation of comprehensive financial disclosures play critical roles. By proposing a model that adeptly leverages this diverse spectrum of textual information, our reinforcement learning framework is uniquely positioned to capture a more holistic view of the factors impacting stock price movements. This integrated perspective aims to enhance the model’s predictive accuracy and, ultimately, its trading performance. The proposed model is shown in Fig. 3.

4.1 Encording

In the original Ramit method, the BERT model [3] was employed to tokenize tweets on a word-by-word basis. However, this approach does not adequately capture the relational nuances and contextual coherence within larger texts, such as earnings reports, where the inter-sentence connections are pivotal. To address this limitation and more accurately represent the intricate relationships between sentences in both short- and medium- to long-term texts, we adopt SentenceBERT [20]. This adaptation allows for the processing of entire sentences, thereby preserving the textual context and enhancing the representation of semantic connections. By inputting full sentences into SentenceBERT, we extract embedded representations from the final hidden layer of each sentence. This unified encoding method via SentenceBERT for all types of text data ensures a more coherent and contextually aware analysis, significantly improving the model’s ability to discern and leverage the nuances in both short-term events and longer-term financial disclosures.

4.2 Feature Generation

This section describes our method for obtaining short-term stock features p'_s , and medium- to long-term stock features p'_l .

Short-Term Texts. For the analysis of short-term texts, spanning back τ days from the trading date, we continue to apply the Time-aware Long Short-Term Memory (TLSTM) and attention mechanisms, as delineated in the Ramit method. This approach yields the short-term market feature p'_s , analogous to the p_τ feature vector in the Ramit method, ensuring that our model captures the transient market sentiments and information flows with improved contextual awareness.

Medium to Long-Term Texts. In contrast to short-term texts, medium- to long-term texts, such as earnings reports, are characterized by less frequent publication intervals. This infrequency introduces a temporal discrepancy, denoted as Δt , between the text’s announcement date and the trading day under consideration. To accurately account for the latent impact of this time difference on stock

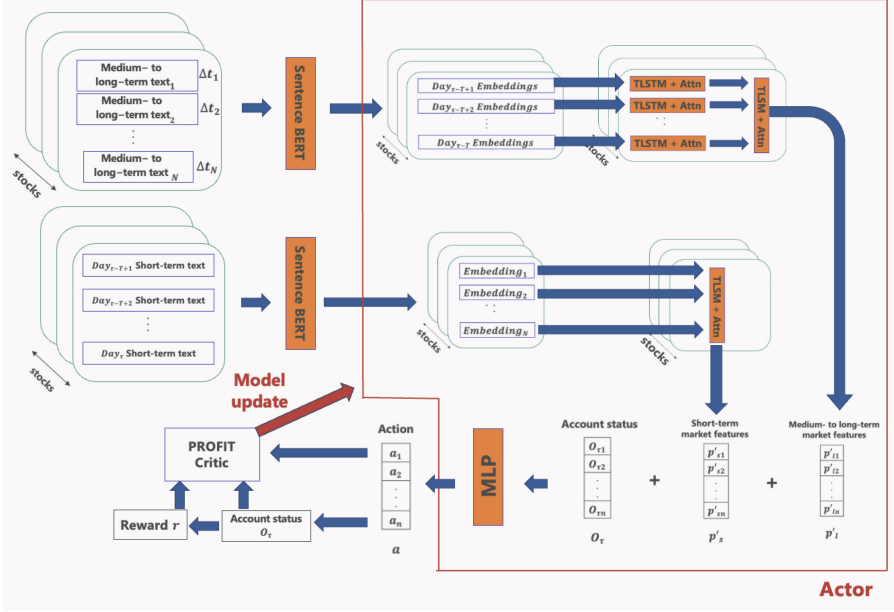


Fig. 3. Proposed method

prices, our model employs Time-aware Long Short-Term Memory (TLSTM) [2]. This approach adeptly captures the nuanced influence of medium- to long-term texts on stock market behavior as it varies over time, ensuring that our analysis reflects the temporal dynamics of the information's market impact.

Furthermore, acknowledging the insight by Barber et al. [1] that not all reports exert equal influence on stock valuation, we implement an attention mechanism within the processing of medium- to long-term texts. This mechanism is designed to sift through the N most recent reports, identifying and emphasizing those with the most significant implications for stock prices. By applying this focused attention layer, we derive the medium- to long-term market feature vector p'_l , which encapsulates the distilled essence and predictive power of the N scrutinized reports.

4.3 Action Determination

To construct a comprehensive state representation at each time step τ , our model aggregates the short-term stock features p'_s , medium- to long-term stock features p'_l , and the trading-account observation vector $O\tau$. These components are concatenated to form the overall stock-level representation $z_\tau = [O_\tau, p'_s, p'_l]$. This holistic state encapsulation ensures that a nuanced understanding of both current market sentiments and deeper financial trends alongside the current trading-account observation informs the decision-making process.

Subsequently, the model employs a Multi-Layer Perceptron (MLP) with a \tanh activation function to determine the trading action a_τ for each stock at this time step. The \tanh function facilitates the output of actions within a normalized range, enabling precise and calibrated trading decisions responsive to the synthesized state information. This methodology allows for the dynamic and informed adjustment of trading strategies, grounded in a comprehensive analysis of both short-term and medium- to long-term market influences.

We update our model by using DDPG, the same as the Ramit method, where PROFIT Critic and Actor are updated using Eqs. (2), (3) and (4), respectively.

5 Experiment

We rigorously train and test our proposed method on 20 of the most actively traded stocks listed on the NASDAQ, one of the leading stock exchanges in the United States. This empirical analysis is designed to comprehensively evaluate our model’s performance and distinct advantages in a real-world trading environment. We conduct a detailed analysis against existing trading support methods to elucidate our approach’s comparative strengths. This comparison aims to highlight our model’s unique contributions and potential improvements, particularly regarding return on investment and risk management. Furthermore, we undertake a series of experiments to assess the robustness and adaptability of our proposed method under varying conditions. By altering the trading period and the initial asset allocation, we aim to explore our model’s operational flexibility. This will allow us to identify the optimal conditions and configurations under which our method exhibits superior performance, thereby providing valuable insights into its practical application and effectiveness in different trading scenarios.

5.1 Datasets

In our experimental design, we leverage a two-pronged approach to text data integration, combining short-term and medium- to long-term textual influences on stock market behavior. For short-term text data, we utilize tweets related to each stock, sourced from the dataset collected by Xu et al.¹, spanning from December 2014 to December 2015. These tweets, all in English and capped at five per stock per date, offer immediate market sentiments and reactions. The dataset is partitioned into a training set covering December 2014 to September 2015 and a testing set, from October 2015 to December 2015, to evaluate our model’s predictive capability over distinct temporal segments.

For medium- to long-term text data, our analysis incorporates both annual (10-K) and quarterly (10-Q) performance reports of each stock, as filed with the U.S. Securities and Exchange Commission², from June 2014 to December

¹ <https://github.com/yumoxu/stocknet-dataset/tree/master>.

² <https://www.sec.gov>.

2015. Due to the varied impact of different sections within these reports, our focus narrows to segments likely to influence stock prices, such as Management’s Discussion and Analysis of Financial Condition and Results of Operations. The temporal division for this data mirrors that of the tweets, with the training period extending from June 2014 to September 2015 and the testing phase from April 2015 to December 2015.

The stock price data, precisely the closing price for each stock on each trading day, is a critical component of our experimental dataset. This consistent use of closing prices in training and testing phases ensures a standardized measure of market response to the synthesized insights derived from our textual analysis.

5.2 Experimental Setup

Trade Conditions. In this study, we focus on a curated selection of 20 stocks from the NASDAQ, specifically targeting those with the highest trading volumes. To incorporate a realistic trading cost into our model, we have standardized the commission rate across all transactions. We assume a commission rate of 0.495% of the total transaction amount, acknowledging the variability of commission structures across different brokerage services. Furthermore, our experiment operates under the constraint that the account balance cannot become negative. Consequently, any attempted purchase exceeding the available account balance is automatically canceled, with the model defaulting to a ‘hold’ action for that trading period.

Hyperparameters. In aligning our experiment with the established Ramit method [24], we configure the market-information observation O_m to retrospectively include short-term texts from 7 days and a single instance of medium- to long-term text. This temporal configuration ensures a balanced incorporation of immediate market reactions and longer-term financial insights into our model’s decision-making process. For the reinforcement learning model updates, like the Ramit method, we adopt a granular approach by setting the mini-batch size to 1, facilitating real-time learning and adaptation. The buffer size is configured at 25. Over the course of the experiment, we undertake a total of 27,200 training steps, equivalent to 100 epochs. The experiment delineates a clear temporal framework for training and testing: the training period spans from January 1, 2015, to September 30, 2015, and the testing period is from October 1, 2015, to December 31, 2015. In the training phase, our simulation is underpinned by an initial asset allocation of 100,000 USD for the training phase, which provides a standardized baseline from which to assess the model’s performance and the effectiveness of its trading strategies.

Evaluation Metrics. We use the rate of return, Sharpe ratio, and maximum drawdown to evaluate the test results.

The Rate of Return (ROI) represents the percentage change in the value of an investment over a specified trading period. This metric provides a straightforward indication of the investment’s performance by quantifying the percentage

gain or loss relative to the initial capital outlay. It is calculated by comparing the final asset value, b_f , to the initial asset value, b_0 , according to the formula:

$$ROI = \frac{b_f - b_0}{b_0} \quad (5)$$

The Sharpe ratio (SR) measures whether or not a return has been made commensurate with the risk taken in the investment. A higher value indicates higher investment efficiency. The trading profit from the method is R_a , and the profit from the risk-free asset is R_f . Note that the value of R_f is set to 0 in this experiment.

$$SR = \frac{E[R_a - R_f]}{std[R_a - R_f]} \quad (6)$$

The maximum drawdown (MDD) is the maximum rate of asset decline over the trading period, and the smaller this value is, the less risky the trade is. The maximum drawdown MDD is expressed as follows, where b_{max} is the maximum amount of the asset during the trading period, and b_{min} is the minimum amount.

$$MDD = \frac{b_{min} - b_{max}}{b_{max}} \quad (7)$$

We compared the following methods.

- T method: Ramit method using only short-term text.
- S method: Proposed method using only medium- and long-term text.
- TSt method: Proposed method considering the time difference between the transaction date and the mid-to the long-term text announcement date.
- TS method: Proposed method without considering the time difference between the transaction date and the mid-to the long-term text announcement date.

5.3 Experimental Results

The results with a trading period of one month and an initial asset value of 100000 USD are shown in Table 1. The descending order of performance based on rates of return is TSt, TS, S, and T. This hierarchy underscores the superiority of the proposed method (TSt) over the existing and other evaluated methods in terms of generating returns. When assessing risk via maximum drawdown, the methods rank from least to most risk as S, T, TSt, and TS. This ranking indicates that the proposed method (TSt) is associated with the highest risk compared to the other strategies. This observation is pivotal, reflecting the trade-off between risk and return that the proposed method embodies. Regarding the Sharpe ratios, the order is TSt, TS, S, and T. This suggests that the proposed method not only yields the highest returns but also does so with a superior risk-adjusted performance. The Sharpe ratio, a measure of the excess return per unit of risk, highlights the efficiency of the proposed method (TSt) in balancing higher returns against the risks incurred.

Table 1. Experimental Results

| Method | ROI | SR | MDD |
|--------|----------------------|------------------|------------------|
| T | -0.0067 ± 0.0018 | -0.24 ± 0.15 | -3.28 ± 0.08 |
| S | -0.0015 ± 0.0023 | 0.05 ± 0.17 | -3.03 ± 0.08 |
| TSt | 0.0296 ± 0.0043 | 1.96 ± 0.25 | -3.68 ± 0.11 |
| TS | 0.0275 ± 0.0039 | 1.95 ± 0.23 | -3.37 ± 0.09 |

The fact that the TSt and TS methods have higher rates of return than the T and S methods suggests that both short-term and medium- to long-term texts impact stock prices. This suggests that using both short-term and medium- to long-term texts in stock price forecasting leads to higher accuracy than using either one of them. The fact that the TSt method has a higher rate of return than the TS method suggests that the passage of time from the announcement date of the medium- to long-term statement affects the stock price. This suggests that, when using medium- to long-term texts in stock price forecasting, considering the passage of time from the announcement date to the trading day at the same time will improve accuracy.

These insights confirm that the proposed method (TSt) not only excels in achieving higher rates of return but also demonstrates an optimized balance between risk and reward, as evidenced by its Sharpe ratio. Although it incurs more risk, as shown by the maximum drawdown metric, the method’s ability to deliver higher risk-adjusted returns positions it favorably against existing and other comparative methods.

5.4 Comparison with Index-Based Method

To evaluate the efficacy of our proposed TSt method as a trading support mechanism, we analyzed its performance against traditional index-based investing. Specifically, we examined the rate of return for the TSt method, with initial assets set at 10,000 USD and a trading period of three months. This period commenced on October 1, 2015, with the initial date normalized to zero for analytical clarity. Figure 4 illustrates the progression of the TSt method’s rate of return compared to the average rate of change in stock prices for the 20 stocks under consideration. The result revealed that, by the end of the trading period, the rate of return achieved through the proposed TSt method notably surpassed the average rate of change in stock prices of the selected stocks. This outcome underscores the TSt method’s superior profitability when juxtaposed with the benchmark index investment approach.

5.5 Effects of Periods and Initial Assets

To rigorously evaluate the performance of our proposed trading method, we conducted tests across a range of trading periods and initial asset levels. The

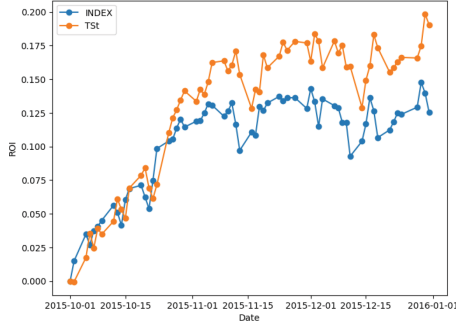


Fig. 4. Comparison with Index Investment

trading periods examined include 1 week, 1 month, 2 months, and 3 months. For each of these durations, we systematically shifted the starting date of the trading period beginning from October 1, 2015, in one-week increments (e.g., to October 8, 2015, October 15, 2015, etc.). Each specific starting date within these increments was tested, and the results were averaged to determine the performance outcome for the respective trading period. Furthermore, we explored the impact of varying initial asset amounts on the efficacy of our trading strategy. The initial assets considered in our tests were 10,000 USD, 100,000 USD, 1,000,000 USD, and 10,000,000 USD. This range was chosen to simulate different scales of investment capital and assess how well our method adapts to varying financial contexts.

Effects of Trading Period. The results are shown in Table 2. In summary, while longer trading periods are conducive to higher profitability, they also introduce greater levels of risk, as evidenced by larger maximum drawdowns. Conversely, shorter trading periods, particularly the one-week timeframe, exhibit a more favorable risk-adjusted return profile, as indicated by their Sharpe ratios. This nuanced understanding of the trade-off between profitability and risk across different trading durations is crucial for tailoring trading strategies to individual risk tolerance and investment objectives.

Effects of Initial Assets. The results are shown in Table 3. The lower initial asset sizes tend to offer higher rates of return and more favorable risk-adjusted returns, as indicated by the Sharpe ratios. Generally, a higher initial investment is expected to lead to a more balanced and potentially conservative strategy, possibly affecting the maximum drawdown and Sharpe ratios.

Table 2. Results in Different trading periods (TSt method)

| Trading period | ROI | SR | MDD |
|----------------|---------------------|-----------------|------------------|
| 1 week | 0.0071 ± 0.0009 | 2.91 ± 0.32 | -0.35 ± 0.02 |
| 1 month | 0.0296 ± 0.0043 | 1.95 ± 0.25 | -3.68 ± 0.11 |
| 2 months | 0.0756 ± 0.0463 | 2.64 ± 1.29 | -4.38 ± 0.54 |
| 3 months | 0.1364 | 2.87 | -5.32 |

Table 3. Results in Different Initial Assets (TSt method)

| Initial asset | ROI | SR | MDD |
|---------------|---------------------|-----------------|------------------|
| 10000.0 | 0.0457 ± 0.0059 | 2.37 ± 0.28 | -4.23 ± 0.12 |
| 100000.0 | 0.0296 ± 0.0043 | 1.95 ± 0.25 | -3.68 ± 0.11 |
| 1000000.0 | 0.0056 ± 0.0011 | 1.88 ± 0.26 | -0.95 ± 0.05 |
| 10000000.0 | 0.0006 ± 0.0001 | 1.87 ± 0.26 | -0.1 ± 0.0 |

6 Conclusion

This paper introduces a novel reinforcement learning approach that synergistically integrates short-term and medium- to long-term text data to construct more informed and robust trading models. Experimental results reveal that the proposed method yields significantly higher returns than existing models and traditional index investment strategies. As we look to the future, this study paves the way for further exploration into integrating even more varied data types, potentially including non-textual information, to refine and enhance predictive models for stock trading across different markets and strategies.

References

1. Barber, B.M., Odean, T.: The effect of attention and news on the buying behavior of individual and institutional investors. *Rev. Financ. Stud.* **21**(2), 785–818 (2007)
2. Baytas, I.M., Xiao, C., Zhang, X., Wang, F., Jain, A.K., Zhou, J.: Patient subtyping via time-aware LSTM networks. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. pp. 65–74 (2017)
3. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805 (2018)
4. Filos, A.: Reinforcement learning for portfolio management. *CoRR*, abs/1909.09571 (2019)
5. Hajime, T., Qiang, M.: Trader characteristics analysis in social trading services (in japanese). In: *DEIM Forum 2017 C7-2* (2017)

6. JSDA. Share distribution survey (in japanese), p. 16 (2023). <https://www.jsda.or.jp/shiryoshitsu/toukei/2022kozinkabunusidoukou.pdf>
7. JSDA. Survey of individual investors' views on securities investment (in japanese), p. 43 (2023). <https://www.jsda.or.jp/shiryoshitsu/toukei/2023kozintoushika.pdf>
8. Kirihata, M., Ma, Q.: Global analysis of factors by considering trends to investment support. In: Hartmann, S., Ma, H., Hameurlain, A., Pernul, G., Wagner, R.R. (eds.) DEXA 2018, Part I. LNCS, vol. 11029, pp. 119–133. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98809-2_8
9. Kirihata, M., Ma, Q.: A trend-shift model for global factor analysis of investment products. *IEICE Trans. Inf. Syst.* **102–D**(11), 2205–2213 (2019)
10. Koratamaddi, P., Wadhvani, K., Gupta, M., Sanjeevi, S.G.: Market sentiment-aware deep reinforcement learning approach for stock portfolio allocation. *Eng. Sci. Technol. Int. J.* **24**(4), 848–859 (2021)
11. Lee, W., Ma, Q.: Whom to follow on social trading services? A system to support discovering expert traders. In: Tenth International Conference on Digital Information Management, ICDIM, pp. 188–193. IEEE (2015)
12. Lee, W., Ma, Q.: Discovering expert traders on social trading services. *J. Adv. Comput. Intell. Intell. Inform.* **22**(2), 224–235 (2018)
13. Lillicrap, T.P., et al.: Continuous control with deep reinforcement learning. In: Proceedings of 4th International Conference on Learning Representations, pp. 159–169 (2016)
14. Liu, Y., Liu, Q., Zhao, H., Pan, Z., Liu, C.: Adaptive quantitative trading: an imitative deep reinforcement learning approach. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 2, pp. 2128–2135 (2020)
15. Liu, Z., Ma, Q.: Unsupervised method for discovering expert traders on social trading services. In: IEEE International Conference on Big Data and Smart Computing, BigComp, pp. 1–8 (2019)
16. Liu, Z., Ma, Q.: Portfolio-based ranking of traders for social trading. *IEEE Access* **8**, 145363–145371 (2020)
17. Ma, C., Zhang, J., Liu, J., Gao, F.: A parallel multi-module deep reinforcement learning algorithm for stock trading. *Neurocomputing* **449**, 290–302 (2021)
18. O'Hara, M.: High frequency market microstructure. *J. Financ. Econ.* **116**(2), 257–270 (2015)
19. Patel, J., Shah, S., Thakkar, P., Kotecha, K.: Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert Syst. Appl.* **42**(1), 259–268 (2015)
20. Reimers, N., Gurevych, I.: Sentence-BERT: sentence embeddings using Siamese BERT-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pp. pp.3980–3990 (2019)
21. Sakai, H., Masuyama, S.: Assigning polarity to causal information in financial articles on business performance of companies. *IEICE Trans. Inf. Syst.* **E92.D**(12), 2341–2350 (2009)
22. Sakaji, H., Sekine, S., Masuyama, S.: Extracting causal knowledge using clue phrases and syntactic patterns. In: Yamaguchi, T. (ed.) PAKM 2008. LNCS (LNAI), vol. 5345, pp. 111–122. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89447-6_12
23. Satoshi, B., Qiang, M.: Hetero-topic model for integrated analysis of stock prices and news(in japanese). In: DEIM Forum 2017 F5-3 (2017)

24. Sawhney, R., Wadhwa, A., Agarwal, S., Shah, R.R.: Quantitative day trading from natural language using reinforcement learning. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 4018–4030 (2021)
25. Sun, S., Wang, R., An, B.: Reinforcement learning for quantitative trading. *ACM Trans. Intell. Syst. Technol.* **14**(3), 1–29 (2023)
26. J. The Investment Trusts Association. Survey results on mutual funds (in japanese), p. 32 (2021). <https://www.toushin.or.jp/statistics/report/research2021/>



TCMIDP: A Comprehensive Database of Traditional Chinese Medicine for Network Pharmacology Research

Lei Ye[✉], Wenxiang Liu, Yuhao Zheng, and Jianhua Li[✉]

East China University of Science and Technology, Shanghai, China
y30221075@mail.ecust.edu.cn, jhli@ecust.edu.cn

Abstract. Network pharmacology is a research method based on biological information data and networks, which can reveal the mechanism of action of Traditional Chinese Medicine, and then discover more active substances with therapeutic effects. However, most of the existing TCM databases lack the collection of TCM prescription data and in-depth data mining and visualization for both TCM and its related information, leading to a limited support in network pharmacology research. In this paper we have constructed Traditional Chinese Medicine Information Database Platform (TCMIDP) for network pharmacology research. It is composed of TCM composition information database and Web-based built-in TCM network pharmacology module. The TCM composition information database collects hierarchical data which contains 4 kinds of TCM resource entities and 6 kinds of associations. In the Web-based TCM network pharmacology module, a visual interactive network diagram displays the data entities and their associations. To mine the TCM data in the above database, node mining and clustering analyses are provided for users to do network pharmacology research. The analyses result, coupled with the visual interactive network diagram can help exploring the 4 types of TCM resource entities that have key regulatory functions in the integrated information network of TCM. TCMIDP makes a significant contribution to data collection, data mining and visualization analysis of TCM, and provides more valuable information support for network pharmacology research.

Keywords: TCM database · Data mining and visualization · TCM network pharmacology research

1 Introduction

Traditional Chinese Medicine (TCM) herbs and prescriptions are increasingly valued for their therapeutic effects and low toxicity in modern medicine, particularly in treating complex diseases like cancer [1, 2]. TCM prescriptions, with their

Supported by Important Drug Development Fund, Ministry of Science and Technology of China (2018ZX09735002).

complex components and synergistic interactions, align with network pharmacology’s “multi-components, multiple targets” approach. Network pharmacology, which analyzes the relationships between diseases and medicines using computational and data management tools, has shifted drug discovery from the “one target, one drug” model to a “network target, multiple component therapeutics” model [3–5]. This advancement is accelerating drug discovery and modernizing TCM.

In recent years, there are a variety of TCM databases and drug target databases [1, 6–9]. But they lack comprehensive prescription data and cross-level associations analysis, hindering in-depth research. Researchers require robust TCM data support and advanced data visualization and mining techniques to fully explore TCM complexities and uncover underlying rules and associations.

In this study, we have built a Traditional Chinese Medicine Information Database Platform (TCMIDP, <http://www.ljh-lab.cn:82/MedicineCore/>), which makes a significant contribution to data collection, data mining and visualization analysis of TCM, and provides more valuable information support for TCM network pharmacology research. Our main contributions include:

- We constructed TCM composition information database(see in Fig. 1A) which collects 4 kinds of entity data of prescriptions, medicinal plants, compounds, targets. Moreover, we modeled and collected 6 kinds of associations(including 3 kinds of direct associations and 3 kinds of cross-level associations) between them. This provides a data base to assist TCM network pharmacology research.
- We transformed the hierarchical integration data into visual network diagrams to visually show the data entities and their associations based on the user’s search result. This provides a powerful tool for TCM network pharmacology research, contributing to a deeper and more comprehensive understanding of the complexity of the TCM system.
- We carried out data mining based on the user’s search result and options to provide more effective information for TCM network pharmacology research and discover potential rules and associations of TCM.

2 Database Contents and Access

TCMIDP models key TCM entities and their associations, providing a comprehensive, accessible resource for understanding TCM efficacy and mechanisms. Through hierarchical data modeling and interactive visualization, it supports advanced research and data mining in network pharmacology.

Hierarchical Data Modeling. In TCMIDP, there are 2 key entities which are prescription and herb, and 2 associated entities which are compound and target. Considering that most herbs come from plants, here medicinal plant is used to stand for herb. There are two steps in data modeling, model the entities and construct the associations between them. The first step is to model the 4

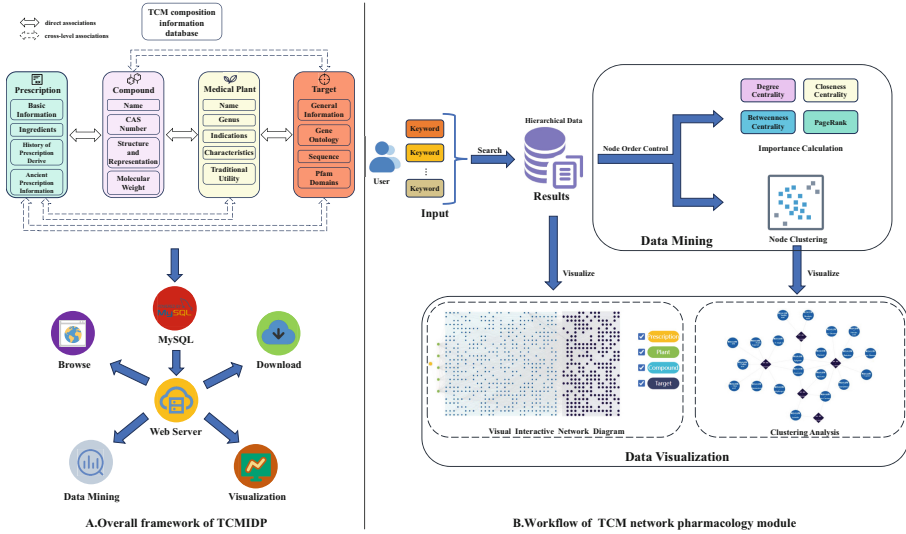


Fig. 1. (A) Overall framework of TCMIDP. (B) Workflow of TCM network pharmacology module.

entities, each with specific information attributes. The second step involves modeling associations between these entities, forming a multi-level chain-associated integration model. Direct associations exist between two entities, while indirect associations form chain-like associations between entities. This model encompasses all data types and associated links relevant to modern research on TCM efficacy and mechanisms, providing foundational data for filtering and mining functionalities.

Database Statistics. TCMIDP contains a TCM composition information database (see in Fig. 1A), encompassing 4 types of TCM resource entities and 6 kinds of associations (Table 1). To improve accessibility, it includes feature ordering and details linking functions. The database comprises 3 direct associations: prescription-plant, plant-compound, and compound-target; and 3 cross-level associations: prescription-compound, plant-target, and prescription-target, facilitating the construction of pathways from TCM resources to the network pharmacology system. The database table structure is shown in Fig. 2.

Result Presentation. TCMIDP uses a visual interactive network diagram (see in Fig. 1B) to enhance data presentation and mining for TCM research. This diagram, constructed using Cytoscape.js, displays hierarchical data through nodes (prescription, plant, compound, target) and edges (associations). Users can interact with the diagram by expanding network relationships, highlighting nodes and their connections, and viewing entity data. This visualization helps users intuitively understand node importance, distribution patterns, and cluster roles in biological networks, supporting effective application of network pharmacology mining algorithms.

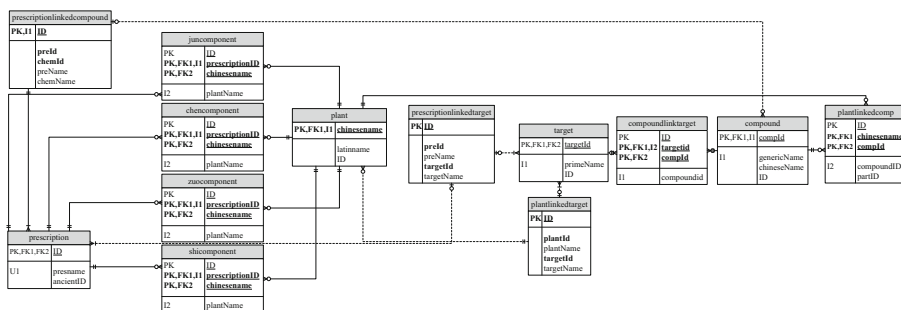


Fig. 2. Database table structure.

Table 1. Data volume and source of TCMIDP

| Item | Amount | Source |
|-------------------------|---------|---|
| Prescription | 182 | The classic prescriptions provided by the ECUST, School of Pharmacy |
| Plant | 9,834 | TCM3D (SIMM, CAS), TCMSP |
| Compound | 23,133 | NatureCompound (SMMU), TCM3D (SIMM, CAS), TCMSP, PubChem |
| Target | 4,351 | Drugbank, PharmMapper |
| Prescription - Plant | 9,297 | Ingredients of a prescription |
| Prescription - Compound | 137,967 | Based on Prescription-Plant, Plant-Compound associations |
| Prescription - Target | 575,958 | Based on Prescription - Plant - Compound - Target association |
| Plant - Compound | 49,805 | NatureCompound (SMMU), TCM3D (SIMM, CAS) |
| Plant - Target | 227,293 | Based on Plant - Compound, Compound - Target associations |
| Compound - Target | 35,383 | TCMSP |

3 Data Mining

Using the visual interactive network diagram and node order control method, TCMIDP enables hierarchical data mining. By clustering plants and compounds based on common compounds and targets, users can identify key prescriptions, TCM components, and action targets. This aids in understanding TCM mechanisms at prescription and molecular levels, advancing TCM and network pharmacology research. TCMIDP also provides comprehensive association data statistics and allows users to download node and edge relationship data for further analysis.

Node Order Control Method. We introduce an order control method for visual interactive network diagrams, based on the theory that significant impact

on a central node occurs within three degrees of separation [10]. This method aggregates distant node information in topological networks, allowing users to assess node importance in sub-networks with varying step lengths. The order is relative; for example, prescription nodes interact with plant nodes at 1st order, compound nodes at 2nd order, and target nodes at 3rd order.

Importance Calculation. The visual network diagram calculates the importance of four types of nodes using Betweenness Centrality (BC), Closeness Centrality (CC), Degree Centrality (DC), and PageRank. These metrics highlight significant nodes, ranked in descending order for easy identification of the top nodes. The backend normalizes and returns results, displayed in a sortable table using abbreviations BC, CC, and DC for simplicity. A counter component controls the recommendation order, from first to third-order subgraphs.

Node Clustering Analysis. We proposed an algorithm to study node correlation within the same hierarchical level by analyzing cross-interactions of components, used for node clustering calculations across four data types: prescriptions, plants, compounds, and targets. Higher proportions of shared components within same-level data indicate greater similarity, such as prescriptions sharing common plants. The algorithm employs both forward and reverse clustering: forward clustering uses the current node and subsequent neighboring nodes for clustering, while reverse clustering uses preceding neighboring nodes. For example, prescriptions are clustered based on shared plants (forward), and plants based on shared prescriptions (reverse). Visualization of clustered nodes, with central nodes as diamonds and clustered nodes as circles, helps observe node distribution and cross-cluster relationships (see in Fig. 1B). Distinct clustered distributions suggest loosely integrated clusters, while non-distinct distributions indicate associations between clusters. Here are two scenarios: If the visualized nodes exhibit a distinct clustered distribution, it indicates that the clustering results are not tightly integrated (see in Fig. 3A); If the visualized nodes do not show a clustered distribution, it suggests a association between the clustering results (see in Fig. 3B).

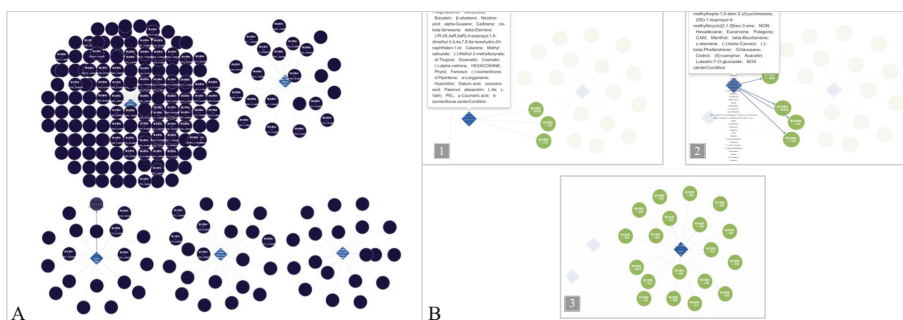


Fig. 3. In Figure A, multi-cluster clustering results are not closely related, There are no related nodes between the five clusters. In Figure B, multi-cluster clustering results are related, cluster 1 and 2 have 3 related nodes, cluster 2 and 3 have 4 related nodes, cluster 1 and 3 have 3 related nodes.

4 User Evaluation

We asked some users about their experience with TCMIDP, and they believe that TCMIDP performs well in terms of data integrity, visualization, data mining capabilities, usability, and interactivity, making it a recommended research tool. However, there are also some shortcomings, such as large data volumes potentially causing slow loading speeds, and some users may need more detailed operational guidance. In future development, optimizations and improvements can be made in these areas to meet the needs of more users.

5 Conclusion

In this work, we have constructed TCMIDP, a platform that serves as a valuable tool for conducting TCM network pharmacology research. To address the limitations of previous TCM databases, we constructed the TCM composition information database and carried out in-depth data mining and visualization. TCMIDP significantly enhances the landscape of data gathering, data mining, and visualization analysis in the field of TCM, offering invaluable information support for advancing TCM network pharmacology research.

References

1. Xue, R., Fang, Z., Zhang, M., Yi, Z., Wen, C., Shi, T.: TCMID: traditional Chinese medicine integrative database for herb molecular mechanism analysis. *Nucleic Acids Res.* **41**(D1), D1089–D1095 (2012)
2. Jin, Y., Zhu, F., Li, J., Ma, L.: TCMFVis: a visual analytics system toward bridging together traditional Chinese medicine and modern medicine. *Vis. Inform.* **7**(1), 41–55 (2023)
3. Hopkins, A.L.: Network pharmacology: the next paradigm in drug discovery. *Nat. Chem. Biol.* **4**(11), 682–690 (2008)
4. Li, S., Fan, T.-P., Jia, W., Lu, A., Zhang, W., et al. Network pharmacology in traditional Chinese medicine (2014)
5. Li, S., Zhang, B.: Traditional Chinese medicine network pharmacology: theory, methodology and application. *Chin. J. Nat. Med.* **11**(2), 110–120 (2013)
6. Chen, C.Y.-C.: TCM database@ Taiwan: the world’s largest traditional Chinese medicine database for drug screening in silico. *PLoS ONE* **6**(1), e15939 (2011)
7. Huang, L., et al.: TCMID 2.0: a comprehensive resource for TCM. *Nucleic Acids Res.* **46**(D1), D1117–D1120 (2018)
8. Ru, J., et al.: TCMSP: a database of systems pharmacology for drug discovery from herbal medicines. *J. Cheminform.* **6**, 1–6 (2014)
9. Wu, Y., et al.: SymMap: an integrative database of traditional Chinese medicine enhanced by symptom mapping. *Nucleic Acids Res.* **47**(D1), D1110–D1117 (2019)
10. Christakis, N.A., Fowler, J.H.: *Connected: The Surprising Power of Our Social Networks and How They Shape Our Lives*. Little, Brown Spark (2009)

Graph Theory and Network Analysis



Fast Subgraph Search with Graph Code Indices

Naoya Funamoto and Akihiro Inokuchi^(✉)

Graduate School of Science and Technology, Kwansai Gakuin University, Sanda,
Hyogo, Japan
{ihn04482,inokuchi}@kwansai.ac.jp

Abstract. The subgraph search problem is of fundamental importance in the fields of information science and database management. In this paper, we propose an index-based subgraph search method that is as fast as the current state-of-the-art technique. The proposed method is an extension of CodeTree, which is a supergraph search method that uses neither enumeration nor graph mining. The extended CodeTree_{sub} treats graphs as graph codes and uses the prefix tree for these graph codes as an index. This index permits the highly efficient filtering of non-solutions, but its construction entails little computational overhead. CodeTree_{sub} effectively limits the number of candidate solutions so that only induced subgraphs of graphs in databases are included in the index, thus accelerating the filtering step. Additionally, CodeTree_{sub} can identify some solutions during the filtering stage. The result is a scalable, high-speed graph filtering and verification method. We compared the performance of CodeTree_{sub} with that of two non-index-based techniques on six benchmark datasets. The results demonstrated that the proposed method was consistently as fast as or faster than the state-of-the-art VEQ_S method in terms of query processing. This study is of particular interest because it illustrates that index-based methods have the potential to outperform non-index-based techniques, thereby providing enhanced query speeds for small- and large-scale databases alike.

1 Introduction

A graph is a data structure that represents objects and the relationships among them. For example, atoms and chemical bonds in molecules may correspond to vertices and edges in graphs, respectively, which allows molecules to be represented as graphs. Additionally, when proteins and the interactions among them correspond to vertices and edges, protein–protein interactions can be represented as graphs. Many objects can be represented in the form of graphs, such as human relational networks, hyperlink structures, and function calls in computer programs. When such objects are represented by graphs and stored in databases, searching for some desired graphs becomes an essential technology in the field of information science. Various graph search techniques exist, such as finding a graph with a perfect match [5, 6], a graph with a matching substructure [12, 18],

or a graph with a similar structure [16, 25]. The subgraph isomorphism problem is NP-complete; hence, solving the subgraph search problem requires the design of efficient algorithms.

Algorithm 1: IFV Procedure

Input : query graph q and index \mathcal{I}

Output: solutions
 $S = \{g_i \in G \mid q \text{ is a subgraph of } g_i\}$

- 1 $P(q) \leftarrow$ Decompose q into a set of patterns
- 2 $Can \leftarrow \bigcap_{p \in P(q)} lookup(\mathcal{I}, p)$
- 3 $S \leftarrow \emptyset$
- 4 **for** $g \in Can$ **do**
- 5 **if** $verification(g, q) = true$ **then**
- 6 $S \leftarrow S \cup \{g\}$
- 7 **return** S

Algorithm 2: vcFV Procedure

Input : set of graphs
 $G = \{g_1, g_2, \dots, g_n\}$
and query graph q

Output: solutions
 $S = \{g_i \in G \mid q \text{ is a subgraph of } g_i\}$

- 1 $S \leftarrow \emptyset$
- 2 **for** $g_i \in G$ **do**
- 3 $\mathcal{A} \leftarrow filter(g_i, q)$
- 4 **if** $\mathcal{A} \neq null$ **then**
- 5 **if** $verification(g_i, q, \mathcal{A}) = true$ **then**
- 6 $S \leftarrow S \cup \{g_i\}$
- 7 **return** S

Algorithm 1 outlines the typical indexing–filtering–verification (IFV) method [18] for the subgraph search problem. In advance of receiving queries, IFV constructs an index \mathcal{I} for a set of graphs G using an enumeration technique. Given a query q , IFV decomposes q into a set of patterns $P(q)$ and obtains $G_S(p) = \{g_i \in G \mid p \text{ is a subgraph of } g_i\}$ with $lookup(\mathcal{I}, p)$ for each pattern $p \in P(q)$. The intersection of sets $G_S(p)$ contains the candidate solutions $Can \subseteq G$. For each candidate $g \in Can$ and q , IFV verifies the subgraph isomorphism problem on Line 5.

Although various IFV-based methods for solving the problem have been proposed since 2000, current mainstream methods are index-free techniques such as CFQL [18] and VEQ_S. In the paper in which VEQ_S was proposed [12], the authors state the following:

Based on our empirical study, building an existing index and filtering using the index incur considerable overhead without gaining higher filtering power for most queries, which is already confirmed by [18]; indeed, the state-of-the-art subgraph search algorithm CFQL [18] has shown that existing indexing methods followed by recent preprocessing and enumeration techniques are inefficient in query processing on widely-used datasets such as PDBS, PCM, and PPI.

CFQL and VEQ_S are based on the vertex connectivity-based filtering–verification (vcFV) framework, the outline of which is presented in Algorithm 2. Given a query q , vcFV constructs an auxiliary data structure \mathcal{A} for q and each graph $g_i \in G$. If \mathcal{A} is not constructed, g_i cannot be a solution; otherwise, vcFV

solves the subgraph isomorphism problem for g_i and q with \mathcal{A} . Unlike Algorithm 1, in Algorithm 2, the *verification* step uses \mathcal{A} , which greatly reduces the search space for the subgraph isomorphism between q and g_i .

Although the use of indices has been discouraged as a tool for subgraph search methods in recent years, there are advantages to using indices in database searches. For example, in relational databases, balance trees are often used as indices. The use of these trees reduces the computational complexity of searches to $\mathcal{O}(\log n)$, where n is the number of tuples in a database. By contrast, the computational complexity of Algorithm 2 is proportional to the number of graphs. Therefore, it is necessary to review the use of indices in graph databases. In this paper, we propose an index-based method for the subgraph search problem. The characteristics of the proposed CodeTree_{sub} method are as follows:

1. non-solution graphs are filtered with high efficiency using indices,
2. indices are constructed without considerable overheads, and
3. the high speed and high filtering performance result in short search times.

2 Preliminaries

A labeled graph is represented as $g = (V, E, \ell)$, where V is a set of vertices, $E \subseteq V \times V$ is a set of edges, and $\ell : V \cup E \rightarrow \Sigma$ is a function for assigning labels Σ to the vertices and edges. In this paper, we express the vertices and edges of g as $V(g)$ and $E(g)$, respectively. Given two graphs $g = (V, E, \ell)$ and $g' = (V', E', \ell')$, if there is an injective function $\phi : V \rightarrow V'$ that satisfies $\forall v, u \in V$, then g is called a subgraph of g' , which is denoted by $g \sqsubseteq g'$:

- $\ell(v) = \ell'(\phi(v))$
- $(\phi(v), \phi(u)) \in E'$ if $(v, u) \in E$
- $\ell((v, u)) = \ell'((\phi(v), \phi(u)))$.

Additionally, if $(\phi(v), \phi(u)) \in E'$ iff $(v, u) \in E$ is also satisfied, then g is called an induced subgraph of g' , which is denoted by $g \sqsubseteq_i g'$. The problem of whether $g \sqsubseteq g'$ is called the subgraph isomorphism problem. This problem is NP-complete.

Given graphs $G = \{g_1, g_2, \dots, g_n\}$ and query graph q as input, the problem we address in this paper is to output a set of solutions $S = \{g_i \in G \mid q \sqsubseteq g_i\}$.

We propose a method based on IFV. The basic idea of IFV-based methods is that, if $p \sqsubseteq q \wedge p \not\sqsubseteq g_i$, then $q \not\sqsubseteq g_i$ [23]. To use this property, IFV computes whether $p \sqsubseteq g_i$ for various patterns p in advance and stores $G_S(p) = \{g_i \in G \mid p \sqsubseteq g_i\}$ for each p . Then, by computing $Can = \bigcap_{p \in P} G_S(p)$ for a set of patterns P , each of which is a subgraph of q , the graphs in G that are not solutions are filtered out, and a set of candidates $Can \subseteq G$ is obtained. Finally, the subgraph isomorphism problem between $g \in Can$ and q is solved. The index \mathcal{I} holds the set of patterns and $G_S(p)$ for each pattern p . In this paper, we discuss the design of such an index.

3 Related Work

Methods based on enumeration techniques [1, 7, 13, 15, 17, 19] exhaustively enumerate all possible patterns in graphs G and store them in an index. The variety of patterns is huge; hence, the size of the index becomes enormous and significant amounts of memory space are required to construct the index. Therefore, this type of method generally limits the number of patterns to simple structures such as paths, cycles, and trees. For example, GraphGrep [17], GraphGrepSX (GGSX) [1], GRAPES [7], and SING [15] enumerate paths from graphs G , whereas CT-Index [13] enumerates cycles. The number of patterns is restricted by limiting the number of vertices or edges within each pattern.

Methods based on graph mining search for subgraph patterns that occur frequently in G , and construct indices from these mined patterns [3, 4, 20, 23, 24]. The support of each pattern p is defined as $sup(p) = |\{g_i \in G \mid p \sqsubseteq g_i\}|$. For a given threshold σ , frequent subgraph patterns in G are $\{p \mid sup(p) \geq \sigma\}$ [10]. In addition to the support, other methods exist for selecting patterns by measuring the filtering ability of the patterns. For example, methyl groups and benzene rings are present in many organic compounds, so they are not always suitable for proper filtering. gIndex [22] uses the discriminative ratio for selection. Mining-based methods require thresholds to be applied to the support or discriminative ratio. It is sometimes difficult to adjust these thresholds, which makes it necessary to repeat the index construction process when they are changed, which entails a long computation time.

Methods based on enumeration and mining are time-consuming for index construction, which makes them ineffective for filtering. Hence, methods based on vcFV without indexing have been proposed in recent years. CFQL [18] constructs an auxiliary data structure called the compact path-index (CPI) between q and $g_i \in G$ during the preprocessing stage, and then performs a verification step with GraphQL [8]. CPI is a spanning tree of the query graph, and each node in the tree has candidate vertices in g_i that may correspond to the node. CPI removes false positive candidates for the node of the query graph and can also determine the most efficient matching order between the vertices in two graphs. By contrast, VEQ_S [12] searches for matching between two graphs. It generates more compact auxiliary data structures between q and g_i than CPI. In this process, the search space is reduced by skillfully handling the neighbor equivalence class among all degree-one vertices in q . Additionally, VEQ_S checks whether two children of each node in the search tree are equivalent using this data structure and prunes the redundant search subspace. The index and vertex connectivity-based filtering-verification framework [18] performs a subgraph search by applying vcFV after filtering non-solutions with indices.

Although we address the subgraph search problem in this paper, we should also discuss the related supergraph search problem, which attempts to find $\{g_i \in G \mid g_i \sqsupseteq q\}$ for some given G and q , [2, 9, 11, 14]. Methods based on indices with enumeration and mining techniques form the bulk of supergraph search techniques, although index-free methods have been proposed recently [11]. Additionally, CFQL and VEQ_S can solve the supergraph search problem by replacing q and g_i on Lines 3 and 5 in Algorithm 2.

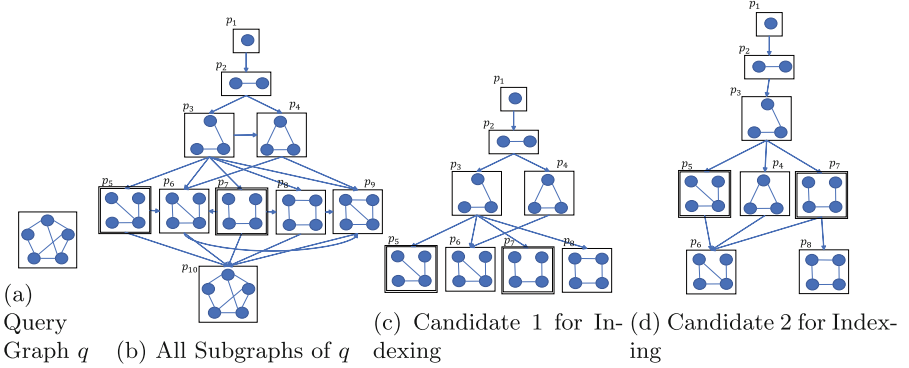


Fig. 1. Relationships between the Query Graph and its Connected Subgraphs

4 Basic Concept of the Proposed Method

Next, we explain the basic concept of indexing in the proposed method. Figure 1(a) shows an example of query graph q . All connected subgraphs $P_c = \{p_1, p_2, \dots, p_{10}\}$ of q are depicted in Fig. 1(b). The graphs are connected by arrows from p_i to p_j in one of the following cases:

- $p_i \sqsubset p_j \wedge |V(p_i)| + 1 = |V(p_j)|$ or
- $p_i \sqsubset p_j \wedge |E(p_i)| + 1 = |E(p_j)|$.

After decomposing q into subgraphs, we obtain sets of graphs $\{g_i \in G \mid p \sqsubseteq g_i\}$ with $lookup(\mathcal{I}, p)$ in Algorithm 1. By intersecting these sets, we limit the candidate solutions to $\bigcap_{p \in P_c} lookup(\mathcal{I}, p) = \bigcap_{p \in P_c} \{g_i \in G \mid p \sqsubseteq g_i\}$. In the case in which the index stores all possible connected graphs, huge amounts of time and memory are required to construct and hold the index [19]. Therefore, it is not practical to store the graphs in the index. Hence, for example, we assume that p_9 and p_{10} are not stored in the index.

To efficiently traverse an index that does not store all possible graphs, we introduce the following lemma. We omit proofs in this paper because of the space limitation.

Lemma 1. *Given query graph q and two patterns p_i and p_j such that $p_i \sqsubset p_j \sqsubseteq q$, candidate solutions for q are included in*

$$\{g \in G \mid p_i \sqsubseteq g\} \cap \{g \in G \mid p_j \sqsubseteq g\} = \{g \in G \mid p_j \sqsubseteq g\}. \quad \blacksquare \quad (1)$$

According to Lemma 1, applying $lookup(\mathcal{I}, p)$ with a larger pattern results in more effective filtering. The maximum patterns among $P'_c = P \setminus \{p_9, p_{10}\} = \{p_1, p_2, \dots, p_8\}$ are p_6 and p_8 . Patterns p_5 and p_7 , enclosed by the double-line square, are not induced subgraphs of q . When p' is not an induced subgraph of q , it is possible that p'' , which contains p' as a subgraph and is an induced subgraph of q , will be stored in the index. In this case, according to Lemma 1, p'' is more effective for filtering than p' . When traversing the index for a given q , we must consider how to efficiently reach p_6 and p_8 via the patterns $P'_c \setminus \{p_5, p_7\}$.

We consider two cases and redraw Fig. 1(b). The first case is that $p_i \sqsubset p_j$ for all graphs in P'_c and $|V(p_i)| + 1 = |V(p_j)|$, as shown in the directed acyclic graph (DAG) in Fig. 1(c). The second case is that $p_i \sqsubset p_j$ for all graphs in P'_c and $|E(p_i)| + 1 = |E(p_j)|$, as shown in the DAG in Fig. 1(d). We wish to obtain patterns p such that $p \sqsubseteq q$ by adopting one of the DAGs as an index and traversing the adopted DAG.

Note that the objective is to reach p_6 and p_8 in Fig. 1(c) or 1(d), not to visit all nodes¹. Therefore, it is desirable to visit fewer nodes so that the index traversal is more efficient. For the case of Fig. 1(c), we can reach p_6 and p_8 without visiting patterns that are not induced subgraphs of q , which enables us to reduce the time required to traverse the index. By contrast, for Fig. 1(d), to reach p_8 , it is necessary to pass through node p_7 , which is not an induced subgraph of q ; this increases the time taken to traverse the index. The gIndex method [22] uses an index that is a spanning tree of the DAG in Fig. 1(d). Patterns found in the nodes in this index are represented as a depth-first search code [21]. Note that gIndex may visit nodes with patterns that are subgraphs of q , but are not induced subgraphs of q . By contrast, we aim to design a method for traversing the DAG in Fig. 1(c) without visiting patterns that are not induced subgraphs of q . For this purpose, we use the Apriori-based connected Graph Mining (AcGM) code [10].

5 Graph Representation and Indexing of Databases

To represent graphs, we use the AcGM code.

Definition 1 (AcGM code [10]). *When the vertex IDs $u_1, u_2, \dots, u_{|V|}$ are assigned to the vertices in a graph $g = (V, E, \ell)$, the graph is represented as the adjacent matrix, where subgraphs induced by u_1, u_2, \dots, u_i ($1 \leq i \leq |V|$) are connected. In the matrix, if $(u, u') \in E$, $x_{u,u'} = \ell((u, u'))$; otherwise, $x_{u,u'} = 0$. In this case, the AcGM code of g is expressed as*

$$\text{code}(g, \langle u_1, u_2, \dots, u_{|V|} \rangle) = s_1 s_2 \cdots s_{|V|},$$

$$\text{wheres } s_i = \ell(u_i) x_{1,i} x_{2,i} \cdots x_{i-1,i}.$$

s_i ($1 \leq i \leq |V|$) is called a code fragment. ■

For a given graph, multiple AcGM codes exist for the different ways of assigning vertex IDs. We denote a set of AcGM codes that represent g by $\Omega(g)$ and a graph represented by a code c by $g(c)$.

For a given set of AcGM codes, we define its prefix tree as the Code Tree.

Definition 2 (Code Tree [9]). *The Code Tree consists of a triplet (N, B, r) , where N is a set of nodes, $B \subset N \times N$ is a set of branches, and $r \in N$ is the root*

¹ We use the terms *vertex* and *edge* for a graph, and the terms *node* and *branch* for an index. Additionally, we use the term *CodeTree* to refer to the method for the graph search and the term *Code Tree* to refer to the index for the graph search.

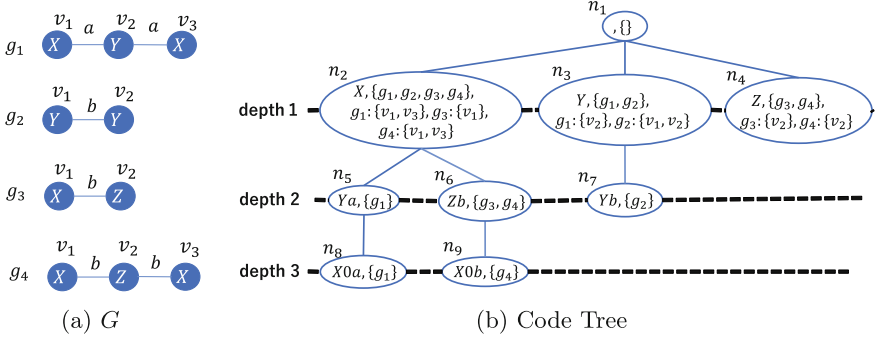


Fig. 2. Example of the Graph Database and Code Tree

of the tree. Each node in the Code Tree has a code fragment and set of graphs. If a code generated by concatenating the fragments associated with the nodes on the path from the root to node n is c and $g(c) \subseteq g_i$ for $g_i \in D$, the set of graphs for n contains g_i . Additionally, each node with the vertex label ℓ at depth 1 has a set of vertices with the label ℓ in each g_i . ■

We denote the code fragment and set of graphs for node n by $fr(n)$ and $G_S(n)$, respectively. Additionally, we denote the graph represented by the code generated by concatenating fragments associated with nodes on the path from the root to node n by $g(n)$. From the Code Tree, we obtain a set of vertices in g_i at node n using $\Lambda(n, g_i)$. For example, for the database that consists of four graphs in Fig. 2(a), one possible code tree is depicted in Fig. 2(b). For n_2 in Fig. 2(b), $fr(n_2) = Y$ and $G_S(n_2) = \{g_1, g_2\}$. Additionally, $\Lambda(n_2, g_2) = \{v_1, v_2\}$.

The Code Tree stores a set of patterns for filtering. Constructing mining-based and enumeration-based indices for the subgraph search requires a huge computation time and significant memory space [18]. Additionally, when the database of graphs is updated, major reconstruction of the indices is required. To avoid these issues, we construct our Code Tree based on the following strategy.

1. We “randomly” generate a connected induced subgraph g_i^s with δ vertices from each $g_i \in G$.
2. We “randomly” generate an AcGM code $c_i \in \Omega(g_i^s)$ of g_i^s .
3. We add c_i such that c_i forms a path from the root of the Code Tree to node n .
4. To filter out graphs with infrequent labels, for each $\ell \in \Sigma$, we create an AcGM code ℓ and apply process (3) to ℓ .

Because only one code is generated from each graph in the database, the time-consuming processes of subgraph mining and enumeration are not required. Despite the simplicity of our indexing method, our subgraph search technique achieves highly efficient filtering and is comparable with VEQ_S in its search processing, as will be demonstrated in evaluation experiments.

Lemma 2. *The number of nodes in the Code Tree and its space complexity are bounded by $\mathcal{O}(|G|\delta)$ and $\mathcal{O}(|G|^2\delta + \sum_{g_i \in G} |V(g_i)|)$, respectively.* ■

Algorithm 3: constructCodeTree

Input : a set of graphs G and the length δ of codes**Output:** a code tree \mathcal{T}

```

1  $\mathcal{T} \leftarrow (\{r\}, \emptyset, r)$ ;
2 for  $g_i \in G$  do
3    $s_1 s_2 \cdots s_\delta \leftarrow getCode(g_i, \delta)$ ;
4    $addPathToTree(s_1 s_2 \cdots s_\delta, \mathcal{T})$ ;
5 for  $\ell \in \Sigma$  do
6    $addPathToTree(\ell, \mathcal{T})$ ;
7  $\mathcal{T} \leftarrow pruningEquivalentNodes(\mathcal{T})$ ;
8 for  $g_i \in G$  do
9    $inclusionCheck(g_i, r, \langle \rangle)$ ;
10 return  $\mathcal{T}$ ;

```

Algorithm 4: inclusionCheck

Input : a graph g_i , node n , and $\langle w_1, \dots, w_h \rangle$ to generate a code from g_i

```

1  $G_S(n) \leftarrow G_S(n) \cup \{g_i\}$ ;
2 if  $depth(n) = 1$  then
3    $V' \leftarrow \{v \mid v \in V(g_i), \ell(v) = fr(n)\}$ ;
4    $add\ "g_i : V'"$  in the node  $n$ ;
5 if  $\nexists m$  s.t.  $m$  is a descendant of  $n$  and  $g_i \notin G_S(m)$  then
6   return;
7  $C \leftarrow \{(w, s) \mid s_1 \cdots s_h s = code(g_i, \langle w_1, \dots, w_h, w \rangle) \text{ is a prefix of } c, c \in \Omega(g_i)\}$ ;
8  $N \leftarrow children(n)$ ;
9 for  $(m, (w, c)) \in N \times C$  do
10 if  $compare(fr(m), c)$  then
11    $inclusionCheck(g_i, m, \langle w_1, \dots, w_h, w \rangle)$ ;

```

The AcGM codes stored in the Code Tree share their prefixes; hence, the actual number of nodes in the tree is much less than $|G|\delta$. By contrast, the number of patterns generated by subgraph mining or enumeration increases exponentially with the number of vertices in graphs contained in the database. Therefore, the index of the proposed method is very compact.

Algorithm 3 contains the pseudocode used to construct our Code Tree \mathcal{T} . Line 1 defines the root of \mathcal{T} . On Line 3, a prefix of length δ (a sequence of δ code fragments) is generated from among the AcGM codes of each graph $g_i \in G$. On Line 4, the prefix is added to the Code Tree to form a path from the root to a node at depth δ . By repeating Lines 5–6 for $|G|$ AcGM codes of length δ and $|\Sigma|$ codes of length 1, the Code Tree (which is our index) is constructed. If there are two or more leaf nodes corresponding to a certain graph, Line 7 of *pruningEquivalentNodes* prunes as many nodes as possible, leaving at least one. At this moment, $G_S(n)$ is empty for each node n . Line 9 finds nodes n that satisfy $g(n) \sqsubseteq g_i$ for each graph $g_i \in G$ and adds g_i to $G_S(n)$. Algorithm 4 generates all

possible AcGM codes from g in a depth-first manner, starting from each vertex in g . Each of the possible AcGM codes c is limited to that for a connected induced subgraph (but not a connected subgraph) of g for which there exist nodes n that satisfy $g(n) \subseteq g(c)$. Generating all possible AcGM codes is equivalent to taking a permutation of the vertices in g_i . However, if the tree is compact, the time required to generate the codes is not significant because the diversity of the codes is limited. The procedures on Lines 5–6 of Algorithm 4 prevent the redundant traversal of a tree that does not update $G_S(n)$. The function *compare* on Line 10 is the same as that in [9].

The characteristics of the Code Tree are as follows:

1. Patterns in the Code Tree are connected induced subgraphs generated at random from $g_i \in G$. The number of patterns is $|G|$.
2. The patterns are included in $g_i \in G$ as induced subgraphs, but not as subgraphs. Graphs included as induced subgraphs have no fewer edges than those included as subgraphs, which is effective for filtering according to Lemma 1.
3. Connected induced subgraphs generated at random from $g_i \in G$ are stored in the Code Tree. No process for selecting the patterns is required.
4. These multiple codes represent each pattern g_i^s , and one of them is selected at random.
5. The number of nodes in the Code Tree is bounded by $\mathcal{O}(|G|\delta)$.

6 Subgraph Search with the Code Tree

In this section, we describe a method for traversing the Code Tree to obtain patterns contained in query q and candidate solutions *Can*, which corresponds to the supergraph search problem. The problem is to output $\{p \in P \mid p \subseteq q\}$ from query q and graphs $P = \{p_1, \dots, p_m\}$, all of which are stored in the index as patterns. Therefore, the pseudocode shown in Algorithm 6 is based on the supergraph search method proposed in [9]. The first characteristics of our proposed method are not only filtering graphs in G but also filtering nodes in each graph in G to reduce the number of nodes in the graph before verification, which reduces the computation time for verification. We call this *node filtering*. Second, when the method visits node n that satisfies $q = g(n)$ while traversing the tree, the graphs in $G_S(n)$ are added to S because they are solutions. When the number of graphs in *Can* has been sufficiently reduced, the computation time for verification is greatly reduced, similar to Lindex+ [23].

Algorithms 5 and 6 contain the pseudocode for the subgraph search based on the above characteristics. Algorithm 5 is based on Algorithm 1. On Line 3, our method traverses the Code Tree to obtain a subset of solutions S and set of candidate solutions *Can*. On Line 2, M is initialized with nodes at depth 1 in the Code Tree. Immediately after Line 3, there are still nodes in M that were not visited in the traversal. Vertices with labels that the nodes have are the target of node filtering that is executed on Line 5. On Line 6, the subgraph isomorphism problem is solved for the node-filtered graph g'_i and query graph q . Lines 10–14 of Algorithm 6 generate code fragments s for connected and induced subgraphs

Algorithm 5: search

Input : query graph q and Code Tree $\mathcal{T} = (N, B, r)$
Output: a set of solution $S = \{g_i \in G \mid q \sqsubseteq g_i\}$

- 1 $S \leftarrow \emptyset, Can \leftarrow G;$
- 2 $M \leftarrow children(r);$
- 3 $traverse(q, r, \langle \rangle, S, Can, M, true);$
- 4 **for** $g_i \in Can \setminus S$ **do**
- 5 $g'_i \leftarrow (V(g_i) \setminus \bigcup_{n \in M} \Lambda(n, g_i), E(g_i) \setminus (V(g_i) \times \bigcup_{n \in M} \Lambda(n, g_i)), \ell);$
- 6 **if** $verification(g'_i, q) = true$ **then**
- 7 $S \leftarrow S \cup \{g_i\};$

8 **return** $S;$

of q and traverse nodes m that satisfy $g(m) \sqsubseteq q$. While traversing the Code Tree, Line 3 filters out graphs that are not solutions. If Can becomes empty, our method backtracks. $mode$ is true if and only if $g(n) \sqsubseteq_i q$ but not if $g(n) \sqsubseteq q$. When our method visits node n that satisfies $q = g(n)$ using the value of $mode$, Lines 1 and 2 add graphs in $G_S(n)$ to S . Lines 6 and 7 prune the search space without changing S and Can according to the following lemma.

Lemma 3. *At node n in the Code Tree, if $S \neq \emptyset$ and $(Can \setminus S) \cap G_S(n) = \emptyset$, S and Can are unchanged at the descendant nodes of n . ■*

7 Experimental Evaluation

7.1 Experimental Settings

We compared the performance of our CodeTree_{sub} with that of GGSX [1], GRAPES [7], VEQ_S [12], and CFQL [18]. We conducted experiments on a machine running an AMD Ryzen Threadripper 3970X 32-Core processor with 128 GB RAM. CFQL and VEQ_S are index-free subgraph search methods that use filtering to construct auxiliary data structures, and then verify the subgraph isomorphism between queries and candidate solutions. GGSX and GRAPES are subgraph search methods based on IFV. They enumerate all paths of length $\delta' = 4$ from graphs in G and then construct indices. We obtained executable files for GGSX, GRAPES, VEQ_S, and CFQL that run on Linux. These were implemented in C++. VEQ_S is the fastest existing subgraph search method.

We implemented our method in Java². We used VEQ_S in our verification, but its source code is not available. Thus, we did the following.

1. We obtained Can by filtering using our method.
2. We wrote the graphs in Can and the query graph to a file.
3. We measured the computation time required by VEQ_S for the file.

² The executable files written in Java and the datasets for evaluation are available at <https://github.com/KG-CodeTree/CodeTree>.

Algorithm 6: traverse

Input : query graph g , node n , $\langle w_1, \dots, w_h \rangle$ to induce a code from g , a set of solutions S , a set of candidates Can , a set of nodes M , and $mode$

- 1 **if** $mode = true \wedge depth(n) = |V(g)|$ **then**
- 2 $S \leftarrow S \cup G_S(n)$;
- 3 $Can \leftarrow Can \cap G_S(n)$;
- 4 **if** $Can = \emptyset \vee \nexists m$ s.t. m is a descendant of n and has not yet been visited **then**
- 5 **return**;
- 6 **if** $S \neq \emptyset \wedge (Can \setminus S) \cap G_S(n) = \emptyset$ **then**
- 7 **return**;
- 8 **if** $depth(n) = 1$ **then**
- 9 $M \leftarrow M \setminus \{n\}$;
- 10 $C \leftarrow \{(w, s) \mid s_1 \dots s_h s = code(g, \langle w_1, \dots, w_h, w \rangle) \text{ is a prefix of } c, c \in \Omega(g)\}$;
- 11 $N \leftarrow children(n)$;
- 12 **for** $(m, (w, c)) \in N \times C$ **do**
- 13 **if** $compare(fr(m), c)$ **then**
- 14 $\lfloor \lfloor traverse(g, m, \langle w_1, \dots, w_h, w \rangle, S, Can, M, mode \wedge (fr(m) = c))$;

The computation time of the proposed method is the time required for the above process, excluding the time required for file I/O. The computation time for verification in the proposed method includes the time required by VEQ_S to construct auxiliary data structures.

Query Sets: Each query graph was generated from $g \in G$ using either a breadth-first search (BFS) or random walk [18]. The specific procedure for generating the query graph is as follows: (1) select graph g at random from G , (2) select vertex v in g at random, (3) add every vertex and edge to the query graph generated by a BFS or random walk starting from v visits, and then (4) return the query when it has visited the predefined number of edges. Each query set $Q_{\epsilon B}$ (BFS) or $Q_{\epsilon R}$ (random walk) consisted of 100 graphs, where $\epsilon \in \{4, 8, 16, 32, 64\}$ represents the number of edges in each query. We call a query set in which the number of edges is small (large) a “small (large) query set”. Because the subgraph search problem is NP-complete, we set a time limit of 10 min to process one query, similar to the experiments conducted by Kim et al. [12]. If the query could not be processed within the time limit, the query processing time (QPT) was recorded as 10 min.

Table 1. Benchmark Datasets

| | $ G $ | $ \Sigma $ | $ V(g) $ | $ E(g) $ | $degree$ | $ \Sigma $ |
|--------|--------|------------|----------|----------|----------|------------|
| AIDS | 40,000 | 62 | 45 | 47 | 2.09 | 4.4 |
| PDBS | 600 | 10 | 2,939 | 3,064 | 2.06 | 6.4 |
| PCM | 200 | 21 | 377 | 4,340 | 23.01 | 18.9 |
| PPI | 20 | 46 | 4,942 | 26,667 | 10.87 | 28.5 |
| IMDB | 1500 | 10 | 13 | 66 | 10.14 | 6.9 |
| REDDIT | 4,999 | 10 | 509 | 595 | 2.34 | 10.0 |

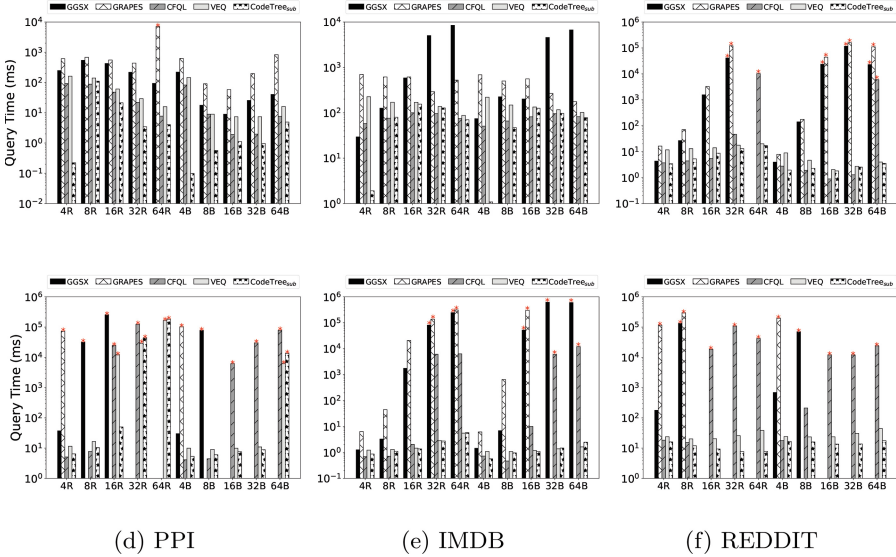


Fig. 3. Query Processing Time on the Benchmark Datasets

Datasets: We used six benchmark datasets (AIDS, PDBM, PCM, PPI, IMDB, and REDDIT), as applied in previous studies [12, 18]. AIDS consists of chemical compounds. PDBS consists of DNA, RNA, and protein structures. PCM and PPI consist of graphs that represent protein–protein interactions; the graphs in PPI are much larger than those in PCM. IMDB is a movie collaboration dataset. REDDIT is a dataset of online discussion communities. IMDB and REDDIT do not contain vertex labels. Thus, one of ten distinct labels was randomly assigned to each vertex [12]. In this paper, we do not provide results for the COLLAB dataset because most methods cannot return solutions for many of the queries within the time limit. Table 1 presents a summary of the datasets. $|V(g)|$ and $|E(g)|$ are the average numbers of vertices and edges of graphs in G , respectively. The *degree* is the average degree and $|\Sigma|$ is the average number of vertex labels in a graph. VEQ_S cannot treat edge labels; hence, we removed edge labels from the datasets.

7.2 Experimental Results

First, we present the results for the QPT, which reflects the core aim of this study. Then we examine detailed results related to query processing and experimental results related to index construction.

Figure 3 shows the QPTs for various query sets on each dataset. A red asterisk indicates that some queries did not yield results within the time limit. If the time limit was exceeded for more than 50 of the 100 queries in each query set, we provided no bar chart for that query set. For each of the six datasets, there were 10 query sets; that is, there were $6 \times 10 = 60$ test cases. In 53 of the 60 cases, the QPTs for CodeTree_{sub} were shorter than those for VEQ_S. Note that we did not count the three cases in which both CodeTree_{sub} and VEQ_S were marked using a red asterisk. CodeTree_{sub} performed well for small queries, and the two datasets AIDS and

Table 2. Search Precision

| Dataset | Query | δ | Search Precision |
|---------|-----------|----------|------------------|
| AIDS | Q_{4R} | 5 | 0.97 |
| AIDS | Q_{4B} | 5 | 0.99 |
| PDBS | Q_{4R} | 20 | 0.59 |
| PDBS | Q_{8R} | 20 | 0.16 |
| PDBS | Q_{16R} | 20 | 0.01 |
| PDBS | Q_{4B} | 20 | 0.72 |
| PDBS | Q_{8B} | 20 | 0.35 |
| PDBS | Q_{16B} | 20 | 0.01 |
| PCM | Q_{8R} | 10 | 0.03 |

REDDIT. The reason that CodeTree_{sub} performed well for small queries is that CodeTree_{sub} found solutions while filtering. Table 2 presents the search precision results for various datasets and query sets. The search precision is the percentage of solutions that the proposed method found while filtering and is defined as $\frac{1}{|Q|} \sum_{q \in Q} \frac{|In(q)|}{|S(q)|}$, where $|In(q)|$ is the number of solutions found by Algorithm 6, but not by Algorithm 5. There are no IFV methods other than CodeTree_{sub} for which search precision is greater than 0. The different values of δ correspond to tuning the QPT of CodeTree_{sub} to be shorter. When the smallest number of vertices in the graphs in query set Q is larger than the depth δ of the Code Tree, the search precision is always zero. We did not include such cases in Table 2. For Q_{4R} and Q_{4B} in the AIDS and PDBS datasets, Algorithm 6 returned many solutions. This is because there were many nodes n up to a depth of 5 in the Code Tree and the graphs $g(n)$ stored in the tree were very diverse. When many solutions were found while filtering, the number of graphs to be verified was greatly reduced and the computation time for verification reduced accordingly. By contrast, the reason that CodeTree_{sub} performed well for the AIDS and REDDIT datasets is that the densities of graphs in the datasets were small. CodeTree_{sub} selected induced subgraph of graphs in G as patterns to be registered in the index. Because the induced subgraph had many edges, the induced subgraph filtered out sparse graphs in G for a given query graph. In [12], it has already been mentioned that CFQL and VEQ_S, which are methods based on vcFV, outperformed GGSX and GRAPES, which are methods based on IFV. For this reason, indices were not used in [18]. However, CodeTree_{sub} is an IFV-based method, and CodeTree_{sub} is as fast as or faster than VEQ_S or CFQL. Therefore, IFV-based subgraph methods are not necessarily slower, and in this paper, we showed that existing methods have room for improvement.

Figure 4 shows the filtering times for the various datasets. The filtering times of CodeTree_{sub} depend on the sizes of the graphs in the query sets and the number of nodes in the Code Tree. When there were few nodes in the Code Tree, the filtering time of CodeTree_{sub} was small because the search space for the queries became narrower, although CodeTree_{sub} filtered out relatively few

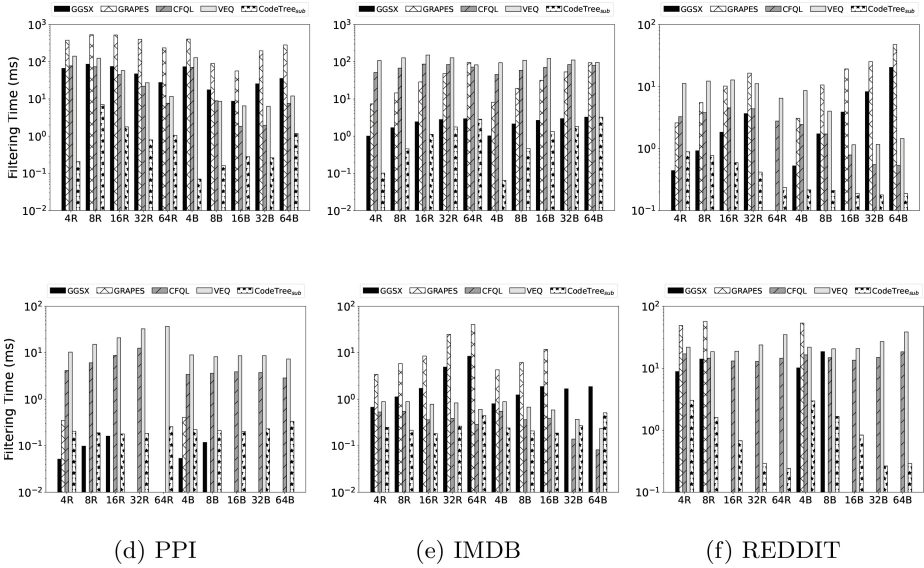


Fig. 4. Filtering Time on the Benchmark Datasets

Table 3. Constructing Indices using IFV-based Methods

| | δ | # of nodes in the Code Tree | Time for constructing index [s] | | | Index size [MB] | | |
|--------|----------|-----------------------------|---------------------------------|------|--------|-------------------------|------|--------|
| | | | CodeTree _{sub} | GGSX | GRAPES | CodeTree _{sub} | GGSX | GRAPES |
| AIDS | 5 | 2,558 | 18.0 | 26 | 12 | 17.1 | 28 | 39 |
| PDBS | 20 | 5,711 | 65.8 | 19 | 4 | 2.6 | 20 | 17 |
| PCM | 10 | 1,555 | 3.2 | 1340 | 233 | 0.6 | 312 | 1360 |
| PPI | 5 | 115 | 0.1 | 6194 | 936 | 0.4 | 23 | 664 |
| IMDB | 4 | 1553 | 0.3 | 57 | 11 | 0.9 | 29 | 38 |
| REDDIT | 3 | 674 | 16.3 | 1645 | 280 | 5.0 | 232 | 1580 |

graphs. As the size of the query graph increased, the filtering time of CodeTree_{sub} became large because the search space for the queries increased as the number of codes generated from the queries increased. By contrast, when the size of the query graph increased, the filtering times of CFQL and VEQ_S decreased. This is because the construction of auxiliary structures involves filtering based on vertex connectivity. Effectively, the vertices in the query have more adjacent vertices, and non-solution graphs can be filtered earlier.

Table 3 presents several details about the construction of indices for IFV-based methods. The different values of δ are the result of tuning the QPT of CodeTree_{sub} to be shorter. The number of nodes in the Code Tree depends on δ , $|G|$, and the characteristics of the dataset. Compared with GGSX and GRAPES, CodeTree_{sub} takes less time to construct indices and requires less memory to hold

the indices. GGSX and GRAPES enumerate all paths of a certain length from the graph in G and store them in indices, which makes the size of their indices larger and their construction time longer.

8 Conclusion

We proposed an index-based subgraph search method that is as fast as the current state-of-the-art technique. CodeTree_{sub} treats graphs as graph codes and uses the prefix tree for these graph codes as an index. This index permits the highly efficient filtering of non-solutions, but its construction entails little computational overhead. CodeTree_{sub} effectively limits the number of candidate solutions so that only induced subgraphs of graphs in databases are included in the index, thus accelerating the filtering step. Additionally, CodeTree_{sub} can identify some solutions during the filtering stage. We compared the performance of CodeTree_{sub} on six benchmark datasets. The results demonstrated that the proposed method was consistently as fast as or faster than the state-of-the-art VEQ_S method in terms of query processing. This study is of particular interest because it illustrates that index-based methods have the potential to outperform non-index-based techniques, thereby providing enhanced query speeds for small- and large-scale databases alike.

References

1. Bonnici, V., et al.: Enhancing graph database indexing by suffix tree structure. In: Proceedings of International Conference on Pattern Recognition in Bioinformatics, pp. 195–203 (2010)
2. Chen, C., et al.: Towards graph containment search and indexing. In: Proceedings of International Conference on Very Large Data Bases, pp. 926–937 (2007)
3. Cheng, J., et al.: FG-index: towards verification-free query processing on graph databases. In: Proceedings of International Conference on Management of Data, pp. 857–872 (2007)
4. Cheng, J., et al.: Efficient query processing on graph databases. *ACM Trans. Database Syst.* **34**(2), 1–48 (2009)
5. Cordella, L., et al.: A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* **26**(10), 1367–1372 (2004)
6. Garey, M., Johnson, D.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. H. Freeman (1979)
7. Giugno, R., et al.: GRAPES: a software for parallel searching on biological graphs targeting multi-core architectures. *PloS One* **8**(10) (2013)
8. He, H., Singh, A.: Graphs-at-a-time: query language and access methods for graph databases. In: Proceedings of International Conference on Management of Data, pp. 405–418 (2008)
9. Imai, S., Inokuchi, A.: Efficient supergraph search using graph coding. *IEICE Trans. Inf. Syst.* **103–D**(1), 130–141 (2020)
10. Inokuchi, A., et al.: A Fast algorithm for mining frequent connected subgraphs. IBM Research Report, RT0448, IBM Research (2002)

11. Kim, H., et al.: IDAR: fast supergraph search using DAG integration. *Proc. of VLDB Endow.* **13**, 1456–1468 (2020)
12. Kim, H., et al.: Versatile equivalences: speeding up subgraph query processing and subgraph matching. In: *Proceedings of International Conference on Management of Data*, pp. 925–937 (2021)
13. Klein, K., et al.: CT-index: fingerprint-based graph indexing combining cycles and trees. In: *Proceedings of International Conference on Data Engineering*, pp. 1115–1126 (2011)
14. Lyu, B., et al.: Scalable supergraph search in large graph databases. In: *Proceedings of International Conference on Data Engineering*, pp. 157–168 (2016)
15. Natale, R., et al.: SING: subgraph search in non-homogeneous graphs. *BMC Bioinform.* **11**(96) (2010)
16. Qin, Z., et al.: GHashing: semantic graph hashing for approximate similarity search in graph databases. In: *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2062–2072 (2020)
17. Shasha, D., et al.: Algorithmics and applications of tree and graph searching. In: *Proceedings of Symposium on Principles of Database Systems*, pp. 39–52 (2002)
18. Sun, S., Luo, Q.: Scaling up subgraph query processing with efficient subgraph matching. In: *Proceedings of IEEE International Conference on Data Engineering*, pp. 220–231 (2019)
19. Williams, D., et al.: Graph database indexing using structured graph decomposition. *Proceedings of IEEE International Conference on Data Engineering*, pp. 976–985 (2007)
20. Xie, Y., Yu, P.: CP-index: on the efficient indexing of large graphs. In: *Proceedings of ACM Conference on Information and Knowledge Management*, pp. 1795–1804 (2011)
21. Yan, X., Han, J.: gSpan: graph-based substructure pattern mining. In: *Proceedings of IEEE International Conference on Data Mining (ICDM)*, pp. 721–724 (2002)
22. Yan, X., et al.: Graph indexing: a frequent structure-based approach. In: *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 335–346 (2002)
23. Yuan, D., Mitra, P.: Lindex: a lattice-based index for graph databases. *VLDB J.* **22**(9), 229–252 (2013)
24. Zhao, P., et al.: Graph indexing: tree + delta \geq graph. In: *Proceedings of International Conference on Very Large Data Bases*, pp. 938–949 (2007)
25. Zhu, Y., et al.: Answering Top- k graph similarity queries in graph databases. *IEEE Trans. Knowl. Data Eng.* **32**(8), 1459–1474 (2020)



Completing Predicates Based on Alignment Rules from Knowledge Graphs

Emetis Niazmand^{1,2}(✉)  and Maria-Esther Vidal^{1,2,3} 

¹ TIB Leibniz Information Centre for Science and Technology, Hanover, Germany
{Emetis.Niazmand, Maria.Vidal}@tib.eu

² Leibniz University of Hannover, Hanover, Germany

³ L3S Research Center, Hanover, Germany

Abstract. Knowledge graphs (KGs) are dynamic structures, often shaped by diverse user communities, leading to the emergence of alternative representations for the same concepts. These alternative definitions, while enriching KGs with complementary information, also pose a challenge for downstream tasks by potentially impeding the completeness of the retrieved information. This paper tackles the problem of identifying alternative definitions of predicates within KGs. We present SYRUP, a method designed to uncover conjunctions of predicates that encapsulate the same semantic relationship as a given predicate but offer complementary instances. Through SYRUP, we aim to augment KG completeness by harnessing these alternative representations. To assess the effectiveness of SYRUP, we conduct an empirical study using a benchmark of 60 SPARQL queries over DBpedia, comprising six distinct domains. Our experimental results demonstrate improvements in both the completeness and correctness of query answers, with accuracy levels ranging from 0.73 to 0.95. Furthermore, we make SYRUP openly accessible on GitHub (<https://github.com/SDM-TIB/SYRUP/>), enabling researchers to replicate our experiments and integrate SYRUP into workflows for KG enhancement.

Keywords: Alternative Definition · Knowledge Graph · Completeness

1 Introduction

Existing knowledge graphs (KGs), such as Wikidata [25] and DBpedia [4], are often constructed using hybrid approaches that combine human intelligence with computational methods. The continuous expansion of entities and properties within these KGs significantly impacts downstream tasks, particularly query processing and question answering. Specifically, a *community-based KG* is a type of KG where multiple individuals or groups from a community collaborate to contribute, curate, and update the KG. This collaborative effort aims to ensure a diverse and comprehensive representation of information. However, contributing communities may introduce predicates with different names referring to the same concepts, or predicates sourced from different vocabularies or

ontologies within the KG. This heterogeneity in predicate nomenclature can lead to issues of redundancy, inconsistency, and incompleteness within the KG. Consequently, downstream tasks such as question answering may fail to retrieve complete answers. Despite the significant efforts made by contributing communities, community-based KGs have the potential to be incomplete (i.e., lacking information about certain entities and properties) due to the decentralized nature of their development and maintenance [2]. Furthermore, adhering to the principles of the open-world assumption (OWA), KGs may inherently be incomplete, meaning that every missing relation between two entities is not assumed to be false, but rather unknown. According to Suchanek et al. [22], between 69% and 99% of entities in KGs fail to include at least one predicate shared by other entities in the same class, highlighting the potential incompleteness of represented relationships when used in downstream tasks.

Focusing on complementary information is essential for enhancing downstream tasks, such as prediction tasks and improving query answer completeness in KGs. Incompleteness within KGs can be addressed by detecting alternative definitions of predicates, thereby contributing complementary instances of these predicates. These alternative definitions play a crucial role in identifying unknown positive facts within KGs. Our approach is to identify incomplete predicates in a KG and to discover complementary conjunctions of predicates. While existing approaches have focused on completing instances of relationships within KGs, to our knowledge, the task of identifying the minimal set of alternative definitions of predicates that are complementary has not been explored. This paper fills this gap by proposing a novel methodology that not only identifies alternative definitions but also demonstrates their effectiveness in enhancing downstream tasks and improving the completeness of query answers in KGs.

Problem Statement: This paper addresses the challenge of completing predicates in KGs by detecting alternative definitions which correspond to conjunctions of predicates, aiming to enhance the completeness of retrieved information. While some alternative definitions may not be complementary and thus have no effect on downstream tasks like query answer completeness, identifying a minimal set of alternative definitions can enhance data retrieval for predicates, enriching the KG with instances that would otherwise be missed.

Proposed Solution: We introduce SYRUP, an engine to identify alternative definitions of predicates. SYRUP implements a two-fold approach that employs a metric called *Complementary Score* to evaluate the impact of alternative definitions on completing predicate $p(., .)$. Additionally, SYRUP leverages ontology and alignment rules to pinpoint a minimal set of alternative definitions for a predicate $p(., .)$. By doing so, SYRUP effectively identifies alternative definitions that can enhance the completeness of instances represented by predicate $p(., .)$.

Evaluation: We assess SYRUP performance in the downstream task of query processing. Based on results reported by Issa et al. [13], about the incompleteness of DBpedia, we aim to determine if the use of alternative definitions of predicates detected using alignment rules, helps to enhance answer completeness. In

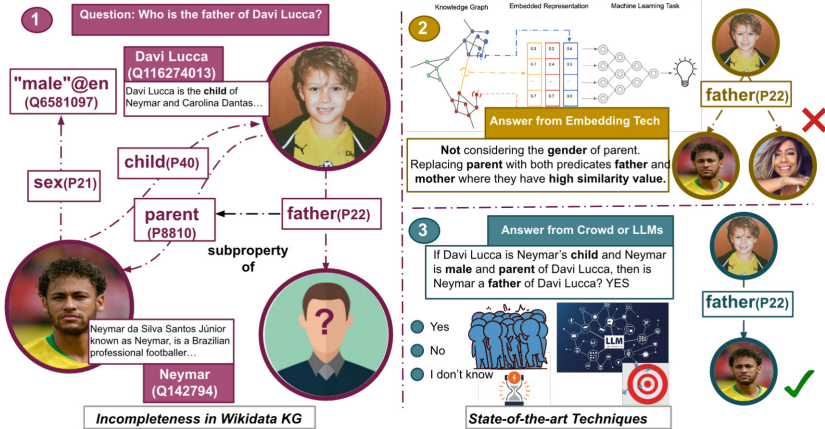


Fig. 1. Motivating Example. 1) Small portion of Wikidata shows incompleteness in the KG to answer the question. 2) Applying embeddings leads to retrieving incorrect answers. 3) Crowd and LLMs are able to answer; but using crowd needs more time and human’s effort; the accuracy of LLMs may not always be optimal. Photos from Wikipedia (<https://de.wikipedia.org/wiki/Neymar>) and Twitter (<https://twitter.com/davilucca99>), (<https://twitter.com/rafabeckranr>). (June 2023)

this case, we only apply the minimal set of alternative definitions during query processing. We created 60 SPARQL queries over six domains from DBpedia (version 2016–10). We measure the performance of SYRUP in terms of the answer completeness and correctness compared with a gold standard created with the help of three annotators. Empirical results show that our approach can enhance answer completeness by increasing the accuracy from 0.73 to 0.95.

Contributions: The main contributions of this paper include: **i)** An approach able to detect alternative definitions of predicates to uncover unknown positive facts in KGs. These methods are implemented in the engine SYRUP. **ii)** *Complementary Score*, a measure to quantify how complementary alternative definitions are. **iii)** An empirical evaluation to validate the effectiveness of the proposed methods. This empirical study is conducted over sixty queries over DBpedia.

This paper is organized into five additional sections. Section 2 motivates our work with an example. Our approach is defined in Sect. 3 with some preliminaries and definitions, and Sect. 4 reports and discusses the results of our empirical study. The related works are briefly described in Sect. 5. Finally, Sect. 6 concludes and outlines our future work.

2 Motivating Example

We motivate our work by providing an example to illustrate the incompleteness of community-based KGs and the existence of alternative definitions of a predi-

cate $p(.,.)$ within KGs that can complete the predicate $p(.,.)$. In Fig. 1, there is a relation as **parent**(.,.), i.e., `wdt:P8810` in Wikidata, between *Davi Lucca* and *Neymar*, but there is no relation between them as **father**(.,.), i.e., `wdt:P22` in Wikidata. Consider a question: “Who is the father of Davi Lucca?”. Due to the incompleteness of Wikidata, the question regarding the **father**(.,.) of *Davi Lucca* returns an empty result (retrieval date, June 2023). Since predicate **parent**(.,.), if and only if parent is a male parent, is an alternative definition of predicate **father**(.,.), then the above question can be answered. Query answer completeness is one of the downstream tasks that can be enhanced by discovering alternative definitions that are complementary to a given predicate in the KG. The state-of-the-art approaches provide different techniques to enhance answer completeness. They address their abilities to capture the semantics of the original KGs using embedding-based techniques. Jain et al. [14] examine whether embeddings are actually able to capture the semantics of the knowledge graph. They demonstrate the weaknesses in semantic representation of embeddings. In our motivating example (Fig. 1), embedding techniques fail to differentiate between predicates **father**(.,.) and **mother**(.,.), as they do not take semantics into account. Therefore, predicate **mother**(.,.) is considered as an equivalent for predicate **parent**(.,.) and subsequently for predicate **father**(.,.), which is not correct. Predicates **mother**(.,.) or **father**(.,.) can be alternative definitions of predicate **parent**(.,.), if and only if it is female parent or male parent, respectively.

The other state-of-the-art approach, HARE [3] exploits crowdsourcing for enhancing the completeness of query answers. Although a crowd can answer the question, it has uncertainty about the output of humans and takes a great deal of time, which means more effort and money. Furthermore, applying Large Language Models (LLMs), such as those underlying ChatGPT¹, can also answer our question; however, an LLM is expensive and may incorrectly answer questions about statements represented in encyclopedic KGs like DBpedia or Wikidata [7]. In this paper, we present an approach for discovering a minimal set of alternative definitions based on alignment rules, which can help us to overcome the incompleteness in KGs [22], and retrieve the complete answers. Thus, we can identify that predicate **parent**(.,.) subsumes predicate **father**(.,.), i.e., all the pairs (X,Y) of entities that satisfy the predicate **father**(.,.), should also satisfy the predicate **parent**(.,.) and Y meets the condition of having **sex**(.,.), i.e., `wdt:P21` in Wikidata, as *male*. As a result, the conjunctive expression **parent**(X, Y), **sex**($Y, "male"@en$) can be considered as an alternative definition of **father**(X, Y). By uncovering the alternative definition of predicate **father**, we can determine the missing relationship between *Neymar* and *Davi Lucca*. So, we conclude that *Neymar* is the **father**(.,.) of *Davi Lucca*. The unknown positive fact in Fig. 1 that corresponds to the instantiations of the pattern $(?x, wdt:father, wd:Davi Lucca)$ can be predicted following the triples $(wd:Neymar, wdt:parent, wd:Davi Lucca)$ and $(wd:Neymar, wdt:sex, "male"@en)$

¹ <https://chat.openai.com/>.

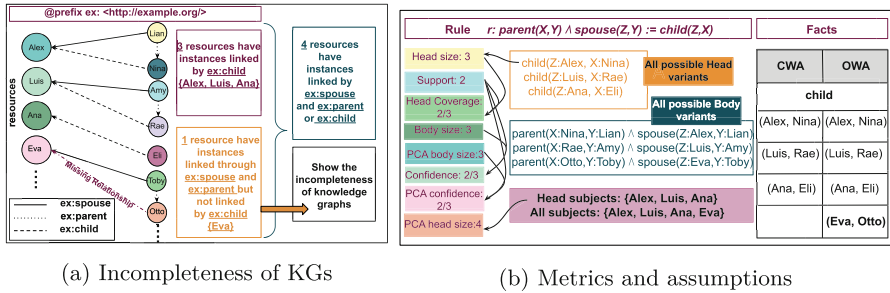


Fig. 2. Open-world assumption (OWA). a) All the couples who are parents do not have child may be caused because of incomplete information. b) Using a metric and following OWA to determine unknown true facts.

which already belong to the KG. Therefore, alternative definitions can infer missing unknown positive facts that are highly likely to be true.

3 The SYRUP Approach

In this section, we formalize the problem of completing predicates by detecting alternative definitions that are complementary for the predicates that miss instances in incomplete KGs. We propose an engine to utilize a metric to quantify the complementarity of alternative definitions; this engine helps uncover unknown positive facts within incomplete KGs and contributes to the enhancement of downstream tasks. Let $KG_{Ideal} = (V, E \cup E', L)$ and $KG = (V, E, L)$ be two KGs. KG_{Ideal} represents an ideal KG comprising all triples from E necessary to form a complete knowledge graph, along with unknown positive triples within E' . The actual knowledge graph (KG) may be incomplete, encompassing only a subset of predicates present in KG_{Ideal} . Notice that because of the open-world assumption (OWA), a KG can comprise the known positive facts; however, there may be unknown positive facts which are not in the KG. In the following, we present the formal definitions related to our approach.

Knowledge Graphs (KGs) [11]. A knowledge graph (KG) is defined as a directed edge-labeled graph, $KG = (V, E, L)$, where V is a set of nodes represented as classes and entities; E is a set of positive facts such as $E \subseteq V \times L \times V$, while the unknown facts correspond to $(V \times L \times V) - E$; and L corresponds to a set of labels. If the KG is expressed in Resource Description Framework (RDF), $t = (s, p, o)$ is a triple in E . Figure 1 shows part of the Wikidata for instances of human, where nodes represent persons with their relationships. The triple $(wd:Neymar, wdt:father, wd:Davi Lucca)$ is one of the unknown positive facts, i.e., unknown facts that are true and should be part of the KG.

Open-world Assumption (OWA) [19]. KGs follow the open-world assumption (OWA); they may be incomplete; the definition of unknown facts is not

precise (Fig. 2a); the unknown facts indicate that the absence of a particular relationship between two entities does not necessarily imply that the relationship is false or non-existent. Rather, it means that the KG does not contain information about that particular relationship [11]. This is different from the closed-world assumption (CWA), where the absence of information implies that the information is false. Therefore, identified alternative definitions based on alignment rules are used to consider the unknown positive facts that are possible incomplete predicates as known positive facts.

Alignment Rules. Alignment rules are logical statements that establish correspondences or mappings between concepts, such as predicates, classes, or entities, either from different ontologies or within the same ontology. In our work, we express these rules as conjunctive Horn clause rules. A Horn clause rule takes the form: $r : B(\bar{T}) := H(X, Y)$, where $B(\bar{T})$, *Body*, is a conjunction of predicate facts (i.e., atoms), and $H(X, Y)$, *Head*, is a predicate fact of a single atom. A rule is considered closed when each variable appears at least twice, ensuring its safety, meaning all variables, i.e., all variables X and Y appear in the set \bar{T} ; Closed rules are always safe, ensuring the validity of entailments. These Horn clause rules represent implications rather than direct equivalences between predicates in the *Body* and the *Head*. **Entailed facts of a rule r** correspond to the instantiations of the predicate fact in *Head* on the substitutions of the variables in *Body* that are positive instantiations of the conjunction of the predicates in *Body*. To evaluate the accuracy and completeness of these rules, we use metrics such as Support, Standard Confidence (SC), and Partial Completeness Assumption (PCA) confidence score, as defined in Galarraga et al. [10]. **Support ($supp(r)$)** quantifies the number of positive entailed facts of the head. Positive entailed facts are the entailed facts belonging to unknown positive facts. $supp(r) := |\{(X, Y) : \exists Z_1, \dots, Z_m : B(\bar{T}) \wedge H(X, Y)\}|$, where Z_1, \dots, Z_m are the variables of the rule apart from X and Y . **Standard Confidence (SC)** measures the ratio of positive predicate facts of the head that are positive entailed facts based on a rule r . **Partial Completeness Assumption (PCA) confidence score:** quantifies the completeness of a KG based on the Partial Completeness Assumption. This assumption states that if a predicate $p(\cdot, \cdot)$ is not functional (i.e., it can have multiple distinct values for the same subject) for each entity s , such as if there exists an instance in a *KG*, then unknown true facts in the *KG* may correspond to known true facts that can be deduced or predicted. These facts are usually named heuristic-based negative edges. Thus, the *PCA* confidence score corresponds to the ratio of $supp(r)$ to the total number of all the predicate facts made by the rule, namely *PCA body size*. *PCA body size* corresponds to the cardinality of the union of positive predicate facts and heuristic-based negative edges of the rule r .

$$PCA(r) = \frac{supp(r)}{|\{(X, Y) : \exists Z_1, \dots, Z_m, Y' : B(\bar{T}) \wedge H(X, Y')\}|} \quad (1)$$

The value of *PCA* confidence is between 0.0 and 1.0. A value of 0.0 shows that none of the entailed instantiations of predicate fact for r belongs to the KG.

A value of 1.0 indicates that all the entailed instantiations of predicate fact for r belong to the KG. Examples of these metrics are shown in Fig. 2b.

Alignment rules can be given as part of the ontology that defines the schema of the KGs, or can be the result of the process of rule mining techniques or other types of learning processes. Given a knowledge graph $KG = (V, E, L)$, a predicate $p(.,.)$ in L , and a set of alignment rules MR over KG where each such alignment rule is an expression of the following particular type: conjunctive rule: $p_1(.,.) \wedge p_2(.,.) \wedge \dots \wedge p_n(.,.) := p'(.,.)$.

Potential Alternative Definitions of a Predicate. We consider a knowledge graph $KG = (V, E, L)$, a predicate $p(.,.)$ in L , and a set of alignment rules MR over KG . A set RT of potential alternative definitions of $p(.,.)$ in KG corresponds to all the rules that have $p(.,.)$ in the head, i.e., rules of the shape $Body := p(X, Y)$ in MR . For simplicity, RT represents each of these rules r as a set of predicates ad that correspond to the predicates in the $Body$ of r .

Example 1. Consider the running example in Fig. 3, the rules in Fig. 3 a) correspond to alignment rules that define the predicate $liveIn(.,.)$. The set RT of potential alternative definitions of $liveIn(.,.)$ comprises five sets, each per rule r including the predicates in the $Body$ of r .

Alternative Definition of a Predicate. Given a knowledge graph $KG = (V, E, L)$, a predicate $p(.,.)$ in L , and RT with the potential alternative definitions of $p(.,.)$ in KG . Let $sim(.,.)$ be a similarity measure and t a threshold for $sim(.,.)$. Let ct be an integer number greater than 0. An alternative definition of $p(.,.)$ is a set of predicates, ad in RT , where the following conditions hold: **Similar predicates**, i.e., there exists at least a p' in ad , such as $sim(p', p) \geq t$; **Complementary instances**, i.e., $|\llbracket p \rrbracket^{KG} \cap \llbracket ad \rrbracket^{KG}| \leq ct$, i.e., the execution of the alignment rule whose by $Body$ corresponds to the predicates in ad differs from the instances of $p(.,.)$ in KG according to a given threshold ct .

Example 2. Given the running example in Fig. 3, the potential alternative definitions are considered as alternative definitions if at least one of the predicates in $Body$ is similar to the predicate $liveIn(.,.)$ (Fig. 3 b) and if there are fewer shared instances between the predicates in $Body$ and the predicate $liveIn(.,.)$ in $Head$. This suggests that the predicates in $Body$ and the predicate $liveIn(.,.)$ in $Head$ represent different instances within the KG, as seen in Fig. 3 c).

Problem Definition. Given a knowledge graph $KG = (V, E, L)$, a predicate $p(.,.)$ in L , and the set RT be a set of potential alternative definitions of $p(.,.)$ in KG . The problem of completing the predicate $p(.,.)$ is defined as the problem of finding a minimal set RT' of alternative definition of $p(.,.)$ that maximizes the deduction of unknown true instances of $p(.,.)$. Formally, the problem is defined as the problem of finding a set RT^* of alternative definition of $p(.,.)$, such that RT^* is a minimal subset of RT and satisfies the following formula:

$$RT^* = \arg \max_{RT' \subseteq RT} [RT']_p^{KG}$$

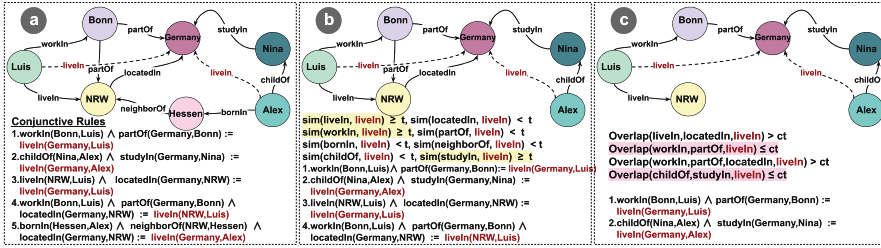


Fig. 3. Running Example. a) Small portion of a KG with the alignment rules associated with it. b) Similarity between at least one predicate in *Body* and predicate in *Head* should be high. c) Overlap between instances of predicates in *Body* and predicate in *Head* should be lower than a given threshold.

where $[RT']_{p^{KG}} = \bigcup_{ad \in RT'} [ad]^{KG} - [p]_{KG}$.

The problem of completing the predicate $p(.,.)$ can be matched to the *Minimum Set Cover* problem, which is known to be NP-hard [17].

Proposed Solution. We propose SYRUP to identify the alternative definitions of a predicate $p(.,.)$ in a knowledge graph KG . SYRUP is a two-fold method composed of two main components. **1)** Create the set RT'' of alternative definitions of $p(.,.)$ according to an input similarity measure $\text{sim}(.,.)$, and two thresholds t and ct . The *Complementary Score* metric is utilized to evaluate the impact of alternative definitions on completing a predicate $p(.,.)$ to increase the number of instances that the predicate $p(.,.)$ misses representing in a KG. **2)** Find the minimal subset RT' that corresponds to a solution to the problem of completing the predicate $p(.,.)$. SYRUP takes as input a knowledge graph KG , a predicate $p(.,.)$, a set MR of alignment rules for the predicate $p(.,.)$, a similarity measure $\text{sim}(.,.)$, and two thresholds t and ct which are calculated by *95th* percentile. The output is a minimal set of alternative definitions of the predicate $p(.,.)$ that can complete $p(.,.)$. SYRUP utilizes $\text{sim}(.,.)$, t , and ct to compute the set of potential alternative definitions of $p(.,.)$. SYRUP resorts to the complementary score (*CompScore*) to traverse the search space and identify the sets of alternative definitions that solve the problem. SYRUP implements a Greedy algorithm to find the minimal subset RT' that maximizes the deduction of unknown true facts of $p(.,.)$. The algorithm iteratively considers alignment rules in RT'' that derive a larger number of unknown true facts of $p(.,.)$, adding rules to RT' accordingly. Ties are broken by randomly selecting one of the tied rules. The algorithm terminates when all rules have been considered, returning RT' . The complementary score (*CompScore*) quantifies the overlap between the answers of the rules added to RT' . The metric $\text{CompScore}(r)$ is quantifying the complementarity of the predicate $p(.,.)$ corresponding to the *Head* of rules in $R = \{r_1, r_2, \dots, r_n\}$. It is computed by evaluating the ratio of the support (*supp*) of a rule r to the maximum value between the *PCA body size* and *PCA head size*. The resulting *CompScore* value falls within the range of 0.0 to 1.0, where a value close to 1.0 indicates that the predicates in the *Body* of a rule can effec-

tively complete the predicate $p(\cdot, \cdot)$ in the *Head*, thus suggesting a high level of complementarity. Conversely, a value closer to 0.0 suggests a lower level of complementarity. This score helps identify whether the predicates in the rule set can be considered as alternative definitions. $CompScore(r)$ is as follows:

$$CompScore(r) = \frac{supp(r)}{\max(|\underbrace{\{\exists Y' : B(\bar{T}) \wedge H(X, Y')\}}_{PCA \text{ body size}}|, |\underbrace{\{\exists X' : B(\bar{T}) \wedge H(X', Y)\}}_{PCA \text{ head size}}|)} \quad (2)$$

As an example, consider the rule $r: parent(X, Y) \wedge spouse(Z, Y) := child(Z, X)$ in Fig. 2b. The $CompScore(r)$ is equal to $\frac{2}{\max(3,4)} = 0.5$.

4 Experimental Study

We assess the performance of SYRUP and compare the observed outcomes with respect to state-of-the-art embedding-based models and the gold standard. This assessment is based on an ideal KG, *KG-Ideal*, completed based on alternative definitions existing in DBpedia in six domains: *Person*, *Music*, *History*, *Film*, *Sport*, and *Drug*. The evaluation aims to formulate the following three research questions: **RQ1**) Can SYRUP detect a minimal set of alternative definitions of predicates using alignment rules from KGs to uncover unknown positive facts? **RQ2**) Can SYRUP enhance query answer completeness over incomplete KGs by detecting unknown positive facts? **RQ3**) Is SYRUP accurately completing query answers using a minimal set of alternative definitions?

Query Benchmark and Gold Standard. We designed a benchmark of 60 queries from six different domains (i.e., Person, Music, History, Film, Sport, and Drug) by analyzing triple patterns answerable by DBpedia (version 2016–10). These domains were chosen due to the varying degrees of incompleteness demonstrated by HARE [3]. We selected queries that do not return all possible correct results because of the incompleteness of DBpedia. The queries have between 2 and 4 triple patterns. The benchmark also comprises the rewritings of the queries resulting from executing query expansion based on the alternative definitions of predicates used in the triple patterns of the original queries. They are considered to evaluate whether they return complete answers by expanding them with minimal number of alternative definitions over DBpedia. We compare our approach to a naive approach that uses all the alternative definitions of predicates in the rewriting process as a baseline. Both baseline and rewritten ones by SYRUP can be found in GitHub². We also established the gold standard for query answer completeness in the benchmark through crowdsourcing; the gold-standard answers correspond to ones obtained if the queries would have been executed over the ideal KG of DBpedia. To achieve this, we extracted predicates from six different DBpedia domains, which were completed by three annotators via Google Forms. Consequently, for each query, the gold standard encompasses

² <https://github.com/SDM-TIB/SYRUP/tree/main/DBpedia>.

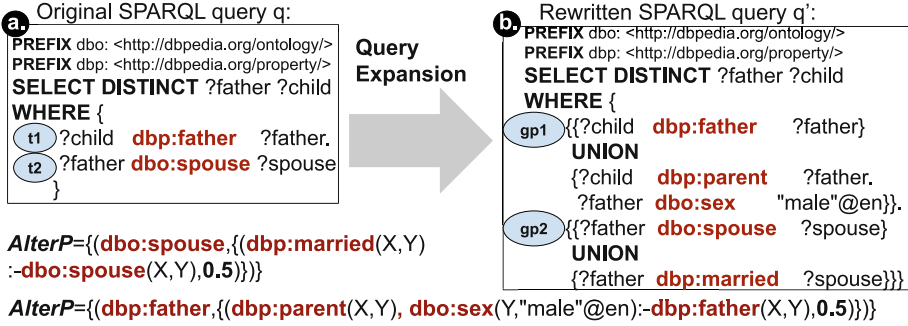


Fig. 4. Example of expanding a SPARQL query using alternative definitions.

all entities that should be part of the answers. Forms and gold standards are accessible at³.

Metrics. To measure the effectiveness of SYRUP, we compute the average of *Percentage of Completeness (PC)* per domain. *PC* corresponds to the ratio of answers produced by baseline ($||[Q]^{Baseline}|$) to the answers of rewritten queries by SYRUP ($||[Q]^{SYRUP}|$), i.e., $PC = \frac{||[Q]^{Baseline}|}{||[Q]^{SYRUP}|} \times 100$. We assess SYRUP accuracy based on *precision* and *recall* computed with respect to the gold standard. *Precision*: The cardinality of the intersection of the correct answers retrieved from the ideal KG and the correct answers retrieved from rewritten queries by detected minimal set of alternative definitions divided by the number of answers retrieved by SYRUP. *Recall*: The fraction of the correct answers retrieved from the ideal KG that intersects with the correct answers retrieved from rewritten queries by detected minimal set of alternative definitions. They are as follows:

$$Precision = \frac{||[Q]^{KG_Ideal} \cap [Q]^{SYRUP}|}{||[Q]^{SYRUP}|} \quad (3)$$

$$Recall = \frac{||[Q]^{KG_Ideal} \cap [Q]^{SYRUP}|}{||[Q]^{KG_Ideal}|} \quad (4)$$

Query Expansion. As an application to validate SYRUP, we use the discovered minimal set of alternative definitions of predicates based on alignment rules to expand the queries of the DBpedia benchmark. Query expansion is a process of rewriting another query that comprises predicates equivalent to the ones included in the original query, but that can enhance answer completeness [24]. Without loss of generality, assume a SPARQL query q comprises a basic graph pattern BGP_q in the *WHERE* clause, and BGP_q is the conjunction of n triple patterns, t_1, t_2, \dots, t_n . Also, assume that the set $P = \{p_1, p_2, \dots, p_n\}$ corresponds to the set of predicates in the triple patterns t_1, t_2, \dots, t_n , such as p_i is the predicate of t_i . Additionally, the set *AlterP* comprises pairs $(p_i, AlterP_i)$ for each p_i in P ,

³ https://github.com/SDM-TIB/SYRUP/tree/main/DBpedia/Gold_Standard.

where $AlterP_i$ is a set of pairs $(r_{i,j}, score_{i,j})$ and $score_{i,j}$ is $CompScore(r_{i,j})$ of rule $r_{i,j}$. Based on query expansion, the rewriting of q is a SPARQL query q' , with the same SELECT clause than q and with a WHERE clause corresponding to a graph pattern GP'_q comprising n graph patterns gp_1, \dots, gp_n connected by the SPARQL JOIN operator. Each gp_i corresponds to a graph pattern composed of the union of the basic graph patterns that correspond to the expansion of the triple pattern $t_i = (s_i, p_i, o_i)$ using alternative definitions of p_i from $AlterP_i$. Figure 4 illustrates the expansion of a SPARQL query based on alternative definitions of both predicates `spouse(.,.)` and `father(.,.)`. As seen, the triple pattern $t1$ is rewritten into the graph pattern $gp1$, comprising the UNION of two basic graph patterns. Alternative definitions are represented with a conjunction of triple patterns (e.g., the second basic graph pattern in $gp1$). The rewritings of the benchmark queries are accessible at⁴.

Implementation. Experiments were run on a Windows 10 machine with an Intel i7-9850H 2.6 GHz CPU and 16 GB 1333 MHz DDR3 RAM. We implemented SYRUP and related metrics in Python 3.10. Queries were executed over a TIB private SPARQL endpoint of DBpedia⁵. To detect alternative definitions, the thresholds t and ct are set up 0.6 and 7, respectively. The alignment rules used by SYRUP engine can be given as part of the ontology that defines the schema of the KGs, or can be the result of the process of rule mining techniques. In this work, the rule mining system AMIE3 [18] is applied to mine the alignment rules; they are extracted on default settings of AMIE. We apply the PCA confidence score and head coverage with a threshold of 0.1 and 0.01, respectively. The maximal number of atoms per rule is 3. We sort the rules first by descending PCA confidence score, and then by descending *Standard Confidence*, and look at the top rules. SYRUP utilizes the Horn clause rules with PCA confidence lower than 1.0. It is important to note that the PCA confidence equal to 1.0 indicates that all the entailed instantiations of predicate fact for r already belong to the KG. Therefore, the alternative definitions are not complementary for predicate $p(.,.)$, and applying them to query answer completeness task has no effect at all. As a result, these alternative definitions cannot be part of the minimal set of alternative definitions in query rewriting to maximize the number of instances.

Precision-Recall Evaluation. We compare the accuracy of SYRUP with the accuracy of some embedding-based techniques such as TransD [15], TransH [26], TransE [5], ComplEx [23], and RDF2Vec [21]; these approaches only are used to detect similar predicates in KGs, while SYRUP detects a minimal number of alternative definitions of predicates. Embedding techniques such as TransE, compute for each triple $t = (s, p, o)$ of a KG, the translation vector from the subject (head entity) to the object of a triple (tail entity) corresponds to the embedding of the predicate (relation). They generate alignment results by calculating the similarity of every property of the first matrix with every property of the second matrix, in the embedding space using similarity measures such as Euclidean distance, cosine similarity, etc. [8]. The precision-recall curves in Fig. 5a indi-

⁴ <https://github.com/SDM-TIB/SYRUP/tree/main/Results>.

⁵ <https://labs.tib.eu/sdm/dbpedia/sparql>.

cate that SYRUP outperforms the embedding techniques. For a recall from 0.3 to 0.7, SYRUP achieves very high precision. This suggests that the answers retrieved from expanded queries by detected minimal set of alternative definitions are closer to the answers retrieved from the ideal KG. Moreover, we can observe that RDF2vec outperforms other embedding techniques such as TransD and TransE. RDF2vec leverages the neighborhood information of entities and properties in KGs to create embeddings [21].

Discussion of the Results. Figure 5a presents the evaluation of computing and comparing *precision* and *recall* values of SYRUP and the state-of-the-art embedding-based approaches. The results suggest that SYRUP discovers unknown positive facts using alignment rules with higher *precision* than embedding-based approaches (**RQ1**). Furthermore, Fig. 5b–Fig. 5g compare the number of answers obtained from original queries and the number of answers from queries rewritten using the SYRUP query expansion technique. The comparison is conducted for each domain of benchmark. The average values of *Percentage of Completeness* (PC) are reported per domain. The results in Fig. 5b–Fig. 5g indicate that the number of answers produced by SYRUP in the majority of cases are higher than the answers of original queries. Also, the average values of *PC* achieved with SYRUP are predominantly greater than 50% indicating that SYRUP enhanced the number of answers of most of the benchmark queries, except for the *History* domain. It is supported by the fact that the *History* domain is less complete than the other domains [3]. For some queries, e.g., *Q9-Person*, *Q1-Drug*, *Q6-Drug*, *Q5-Music*, *Q10-Music* and *Q3-History*, query answers are not enhanced by SYRUP. This means that, those queries are already complete (**RQ2**). The average of *precision* and *recall* of baseline and SYRUP is shown in Table 1 and Table 2, respectively. The computed *precision* and *recall* of the results retrieved from rewritten queries using SYRUP with respect to the ideal KG show that in most of the domains the values of *precision* and *recall* are high. The *precision* and *recall* values in other domains in average are higher than 0.88 and 0.91, respectively. The baseline consistently achieves a precision score of 1.0, as it successfully retrieves answers that exactly match those found in the ideal KG. However, its recall value is relatively low because the baseline’s retrieved answers are not as complete as those in the ideal KG (**RQ3**).

Table 1. Mean precision

| Domains | Baseline | SYRUP |
|---------|----------|-------|
| Film | 1.0 | 0.95 |
| Sport | 1.0 | 0.88 |
| Person | 1.0 | 0.88 |
| Drug | 1.0 | 0.73 |
| Music | 1.0 | 0.95 |
| History | 1.0 | 0.92 |

Table 2. Mean recall

| Domains | Baseline | SYRUP |
|---------|----------|-------|
| Film | 0.61 | 0.99 |
| Sport | 0.76 | 1.0 |
| Person | 0.76 | 0.91 |
| Drug | 0.66 | 0.93 |
| Music | 0.67 | 1.0 |
| History | 0.47 | 1.0 |

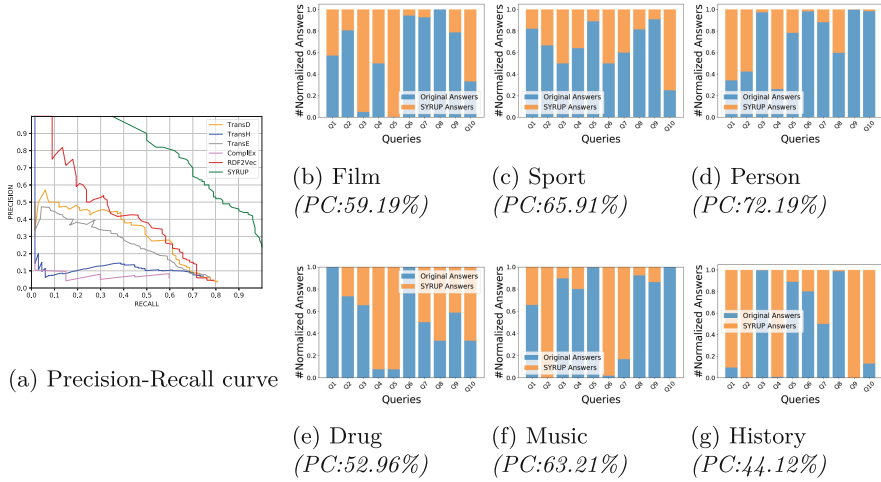


Fig. 5. (a) Experimental results of SYRUP using alignment rules and embedding-based approach for detecting alternative definitions compared with crowdsourcing, (b-g) Size of query answer achieved by original queries and rewritten queries using query expansion by SYRUP per query and domain in DBpedia.

5 Related Work

Despite the availability of many well-performing approaches, there is no research regarding identifying a minimal set of alternative definitions of predicates which complete the instances that are missed by those predicates within KGs. To tackle the problem of completing the instances of a given predicate, several approaches have been employed, including syntactic and semantic methods. Acosta et al. [3] propose a hybrid SPARQL query engine, called HARE, that exploits a micro-task mechanism for enhancing the completeness of query answers. HARE automatically identifies portions of a SPARQL query that might yield incomplete results, and it resolves them via crowdsourcing, which may cost more effort and money. Kalo et al. [16] provide a technique to detect synonymous relationships in large KGs using association rule mining. Abedjan et al. [1] employed frequent item set mining through aggregating positive and negative association rules at the statement level to detect synonym predicates. However, none of them applied discovered synonym predicates to complete the other predicates and increase the number of instances of those predicates. Cheng et al. [6] propose an approach leveraging vocabulary mappings to process queries over federations with heterogeneous vocabularies. However, their experimental study introduces overhead due to the lack of query performance optimization through the application of rewriting rules to logical query plans. To minimize these additional overheads, optimizing query processing can be achieved by applying a minimal set of alternative definitions using alignment rules. To address the problem of incompleteness, researchers have proposed several techniques. Issa et al. provide an

overview in [12], including data augmentation, entity resolution, and knowledge graph enrichment. A study by Galárraga et al. [9] proposed an extension of the SPARQL query language to support completeness. Purohit et al. [20] propose a hybrid method that integrates symbolic and numerical techniques, leveraging the PCA heuristic to capture implicit knowledge and enrich KGs. However, despite the effort conducted by previous works to enhance KG completeness, to our knowledge, no existing approaches have explored the minimal number of alternative definitions of predicates which complete the instances missed by predicates in KGs to achieve this goal.

6 Conclusions and Future Work

This study highlights the problem of discovering alternative definitions of predicates using alignment rules to detect unknown positive facts in KGs. SYRUP follows a two-fold approach to detect a minimal set of alternative definitions for a predicate. The experimental results depict that SYRUP identifies alternative definitions capable of deriving unknown true facts. The evaluation of SYRUP on 60 SPARQL queries over six different domains of DBpedia suggests that SYRUP improves the answer completeness and correctness by increasing the accuracy from 0.73 to 0.95. Expanding queries by the minimal number of discovered alternative definitions from alignment rules is efficient whenever it is performed with respect to the domains, e.g., *Film* and *Music* where there are more alignment rules. In the future, we plan to apply SYRUP to enhance other downstream tasks, e.g., prediction tasks and negative sampling. By applying alternative definitions of predicates, we can enhance the entity’s neighborhood while computing KG embeddings using a particular KG embedding model, especially during negative sampling. It can maximize the probability of observing positive pairs, while minimizing the probability of observing negative pairs.

Acknowledgement. This work is partially funded by EraMed project P4-LUCAT (GA No. 53000015) and the project TrustKG-Transforming Data in Trustable Insights with grant P99/2020.

References

1. Abedjan, Z., Naumann, F.: Synonym analysis for predicate expansion. In: ESWC 2013 (2013). https://doi.org/10.1007/978-3-642-38288-8_10
2. Abiteboul, S.: Querying semi-structured data. In: Database Theory - ICDT '97, vol. 118. https://doi.org/10.1007/3-540-62222-5_33
3. Acosta, M., Simperl, E., Flöck, F., Vidal, M.E.: Enhancing answer completeness of sparql queries via crowdsourcing. SSRN Electron. J. (2017). <https://doi.org/10.2139/ssrn.3199306>
4. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: Dbpedia: a nucleus for a web of open data. In: ISWC 2007 + ASWC (2007). https://doi.org/10.1007/978-3-540-76298-0_52

5. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: NIPS'13 (2013). <https://dl.acm.org/doi/10.5555/2999792.2999923>
6. Cheng, S., Ferrada, S., Hartig, O.: Considering vocabulary mappings in query plans for federations of RDF data sources. In: Sellami, M., Vidal, M.E., van Dongen, B., Gaaloul, W., Panetto, H. (eds.) Cooperative Information Systems. CoopIS 2023. LNCS, vol. 14353, pp. 21–40. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-46846-9_2
7. Dong, X.L.: Generations of knowledge graphs: the crazy ideas and the business impact. CoRR (2023). <https://doi.org/10.48550/arXiv.2308.14217>
8. Fanourakis, N., Efthymiou, V., Kotzinos, D., Christophides, V.: Knowledge graph embedding methods for entity alignment: experimental review. Data Min. Knowl. Discov. (2023). <https://doi.org/10.1007/S10618-023-00941-9>
9. Galárraga, L., Hose, K., Razniewski, S.: Enabling completeness-aware querying in SPARQL. In: WebDB (2017). <https://doi.org/10.1145/3068839.3068843>
10. Galárraga, L.A., Teflioudi, C., Hose, K., Suchanek, F.M.: AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In: WWW (2013). <https://doi.org/10.1145/2488388.2488425>
11. Hogan, A., et al.: Knowledge Graphs (2021). <https://doi.org/10.2200/S01125ED1V01Y202109DSK022>
12. Issa, S., Adekunle, O., Hamdi, F., Cherfi, S.S., Dumontier, M., Zaveri, A.: Knowledge graph completeness: a systematic literature review. IEEE Access (2021). <https://doi.org/10.1109/ACCESS.2021.3056622>
13. Issa, S., Paris, P., Hamdi, F.: Assessing the completeness evolution of dbpedia: a case study. In: ER 2017 Workshops (2017). https://doi.org/10.1007/978-3-319-70625-2_22
14. Jain, N., Kalo, J.C., Balke, W.T., Krestel, R.: Do embeddings actually capture knowledge graph semantics? In: Verborgh, R., et al. The Semantic Web. ESWC 2021. LNCS, vol. 12731, pp. 143–159. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77385-4_9
15. Ji, G., He, S., Xu, L., Liu, K., Zhao, J.: Knowledge graph embedding via dynamic mapping matrix. In: ACL 2015 (2015). <https://doi.org/10.3115/v1/p15-1067>
16. Kalo, J.C., Mennicke, S., Ehler, P., Balke, W.T.: Detecting synonymous properties by shared data-driven definitions. In: Harth, A., et al. (eds.) The Semantic Web. ESWC 2020. LNCS, vol. 12123, pp. 360–375. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49461-2_21
17. Karp, R.M.: Reducibility among combinatorial problems. In: Complexity of computer computations, pp. 85–103. Springer, Boston, MA (1972)
18. Lajus, J., Galárraga, L., Suchanek, F.: Fast and exact rule mining with AMIE 3. In: Harth, A., et al. (eds.) The Semantic Web. ESWC 2020. LNCS, vol. 12123, pp. 36–52. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49461-2_3
19. Loyer, Y., Straccia, U.: Any-world assumptions in logic programming. Theor. Comput. Sci. (2005). <https://doi.org/10.1016/j.tcs.2005.04.005>
20. Purohit, D., Chudasama, Y., Rivas, A., Vidal, M.: Sparkle: symbolic capturing of knowledge for knowledge graph enrichment with learning. In: K-CAP 2023 (2023). <https://doi.org/10.1145/3587259.3627547>
21. Ristoski, P., Rosati, J., Noia, T.D., Leone, R.D., Paulheim, H.: RDF2VEC: RDF graph embeddings and their applications. Semant. Web (2019). <https://doi.org/10.3233/SW-180317>

22. Suchanek, F.M., Gross-Amblard, D., Abiteboul, S.: Watermarking for ontologies. In: Aroyo, L., et al. (eds.) *The Semantic Web – ISWC 2011*. ISWC 2011. LNCS, vol. 7031, pp. 697–713. Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25073-6_44
23. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: *ICML 2016*. <http://proceedings.mlr.press/v48/trouillon16.html>
24. Vechtomova, O.: Query expansion for information retrieval. In: *Encyclopedia of Database Systems* (2018). https://doi.org/10.1007/978-1-4614-8265-9_947
25. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. *ACM* (2014). <https://doi.org/10.1145/2629489>
26. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. *AAAI Press* (2014). <http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8531>



Enriching Hierarchical Navigable Small World Searches with Result Diversification

Mauro Weber¹, João Silva-Leite¹, Lúcio F. D. Santos², Daniel de Oliveira¹,
and Marcos Bedo¹✉

¹ Institute of Computing/UFF – Niterói, Rio de Janeiro, Brazil
{mauro_weber, joaovitorleite}@id.uff.br, {danielcmo, marcosbedo}@ic.uff.br

² Federal Institute of North of Minas Gerais/IFNMG – Montes Claros,
Minas Gerais, Brazil
lucio.santos@ifnmg.edu.br

Abstract. Hierarchical Navigable Small World (HNSW) delivers state-of-the-art performances for approximate k -nearest neighbors (k NN) queries in public benchmarks (e.g., ANN-Benchmarks). While it scales well for large datasets, characterizing the impact of the small-world construction strategy on the search quality is still an open issue. This paper investigates how result diversification can shed light on that question by adding a parameterless strategy to HNSW that explicitly uses local distance-based structures to produce diversified neighbors. Accordingly, we propose (i) a new heuristic for the small-world construction based on the *Influence* concept derived from ball partitioning, and (ii) an extension to HNSW k NN searching algorithm that supports *Influence*-driven result diversification (kN_dN). We evaluated our approach on ANN-Benchmarks, and results show that *Influence*-based partitioning of the search space can substantially enhance the kN_dN quality (Recall by proximity) with a throughput comparable to the standard HNSW construction strategy.

Keywords: k NN · Result Diversification · HNSW · ANN

1 Introduction

The advance of deep learning-based applications has opened the opportunity to produce large and heterogeneous datasets of vector representations from text and images embedded in non-sparse, high-dimensional spaces [1]. Exact search methods based on the Metric Spaces Theory typically struggle to find the k -nearest neighbors (k NN) from a given query object due to the occurrence of the *distance concentration* phenomenon within such datasets [8]. Approximate similarity search allows more efficient data retrieval in those cases by relaxing the constraint of exact distance-based matches. The *Hierarchical Navigable Small World* (HNSW) method provides state-of-the-art performances in the execution of approximate k NN queries in large and high-dimensional datasets [1, 6]. It relies on a hierarchical structure in which every layer is a connected graph where (i) the number of connections for each node is limited by a construction parameter and

(ii) nodes are reachable within a few hops, as in a small world representation [9]. HNSW employs a greedy heuristic to select and rewire connections by adopting a hyperplane-based criterion to skip connections to elements closer to previously inserted neighbors, thus creating long edges based on that separation [6, 7]. Then, the HNSW k NN search algorithm relies on a best-first index traversal while partially sorting the visited nodes in two priority queues [7]. Accordingly, HNSW usage typically demands fine-tuning for construction (connection degree – M) and search (size of priority queues – ef) parameters [1, 9].

While this tuning task is an expected handicap, it may overshadow two relevant questions involving the HNSW characterization, namely **(Q1)** how other partitioning strategies affect the behavior of HNSW?, and **(Q2)** how does HNSW perform when faced with more complex search criteria, such as k NN with result diversification (kN_dN)?. In this paper, we investigate those questions by coupling a ball-partitioning strategy called *Influence* [4] on HNSW. This parameterless strategy partitions the search space using a query object, indexing elements that are *Influenced* by nearest neighbors into ball regions that increase monotonically in coverage. We enforce this criterion to construct the HNSW last layer by using incrementally inserted elements on HNSW as query objects. Then, we extend the HNSW searching algorithm to fetch only non-*Influenced* neighbors as the kN_dN result set. We also implement our proposal on the `nsmlib` library so that we can easily bind it to the public benchmark `ANN-Benchmarks`¹ for comparison with standard HNSW. An experimental comparison showed that partitioning the search space by *Influence* can improve the quality of kN_dN queries.

This paper is organized as follows. Section 2 presents the concepts and related work. Section 3 discusses our implementation. Section 4 presents the experimental evaluation, while Sect. 5 provides the conclusions and future work.

2 Preliminaries and Related Work

Similarity Searching. The core of similarity searching is the distance function (δ) used to measure the proximity between objects, such as $\delta = L_2$, which quantifies the dissimilarity between dimensional vectors. The organization of dataset distances to a fixed object defines a similarity search criterion, such as k NN.

k NN Search. The k -Nearest Neighbors (k NN) query retrieves a set of the k closest elements from a dataset $\mathcal{O} \subset \mathbb{R}^d$ to a reference object $o_q \in \mathbb{R}^d$. Formally, an incremental k NN set, denoted as $k\text{NN}(o_q, \delta, k, \mathcal{O}) = o_1, o_2, \dots, o_k$, is as follows:

$$o_1 = o_i \in \mathcal{O}, \forall o_j \in \mathcal{O}, \delta(o_i, o_q) \leq \delta(o_j, o_q),$$

$$o_{m=2, \dots, k} = o_i \in \mathcal{O} \setminus \cup_{h=1}^{m-1} o_h, \forall o_j \in \mathcal{O} \setminus \cup_{h=1}^{m-1} o_h, \delta(o_i, o_q) \leq \delta(o_j, o_q)$$

Influence Set. The *Influence Set* of a diversified neighbor o_i to a reference o_q (\ddot{I}_{o_i, o_q}) covers each entry o_j of a dataset $\mathcal{O} \setminus \{o_i, o_j\} \subseteq \mathbb{R}^d$ that are (i) farther from o_q than o_i and (ii) more *Influenced* by o_i than o_q , i.e., $\ddot{I}_{o_i, o_q} = \{o_j \mid o_j \in \mathcal{O} \setminus \{o_i, o_q\}, I(o_i, o_j) > I(o_i, o_q) \wedge I(o_i, o_j) > I(o_j, o_q) \wedge I(o_i, o_q) \neq I(o_j, o_q)\}$.

¹ Available at <https://ann-benchmarks.com/>.

kN_dN search. A kNN query with result diversification (kN_dN) retrieves the k non-*Influenced* and nearest elements in $\mathcal{O} \subset \mathbb{R}^d$ to a reference object $o_q \in \mathbb{R}^d$ so that $kN_dN(o_q, \delta, k, \mathcal{O}) = \mathcal{R} = \{o_1, o_2, \dots, o_k\}$ is incrementally defined as follows.

$$\begin{aligned}
 o_1 &= o_i \in \mathcal{O}, \forall o_j \in \mathcal{O}, \delta(o_i, o_q) \leq \delta(o_j, o_q), \\
 o_{m=2, \dots, k} &= o_i \in \mathcal{O}, (\forall o_j \in \cup_{h=1}^{m-1} o_h \Rightarrow o_i \notin \dot{I}_{o_j, o_q}) \wedge (\forall o_g \in \mathcal{O} \setminus \cup_{h=1}^{m-1} o_h \Rightarrow \\
 &\quad (\delta(o_i, o_q) \leq \delta(o_g, o_q) \vee \exists o_j \in \cup_{h=1}^{m-1} o_h \Rightarrow o_g \in I_{o_j, o_q})).
 \end{aligned}$$

Related Work. Graph-based indexes, such as HNSW and NSG, have provided excellent results to speed-up approximated kNN searches [3, 5–7]. Graph structures are particularly suitable to solve kNN searches in high-dimensional spaces, with HNSW delivering consistent performances in *intrinsically* high-dimensional datasets [1, 9]. Such indexes rely on incremental constructions to increase kNN search performance by using *Best-First* search algorithms, which efficiently traverse graphs whose nodes have a limited degree with a greedy, partial sorting principle [7]. Moreover, kNN quality in concentrated datasets can be improved through additional search criteria, such as result diversification [4, 5]. In particular, the proposal in [5] showed significant kNN gains on Gabriel graphs, with similar findings for exact searches on VP-Trees [4]. Accordingly, we explore the HNSW potential to execute kNN queries with result diversification.

3 Material and Methods

A New Ball-partitioning-based Strategy for HNSW. The HNSW construction strategy can be implemented with *ball*-based constraints, producing ball-tree-like partitions that are suitable to execute kNN queries with result diversification [2]. The *Influence*-based partitioning is a natural candidate to extend the HNSW hyperplane-based construction because (i) it relies on dynamic thresholds (*ball radii*) induced by the query locality and (ii) the partitions can be efficiently scanned regarding kNN queries with result diversification [4, 5]. Thus, we propose a new HNSW construction strategy (named

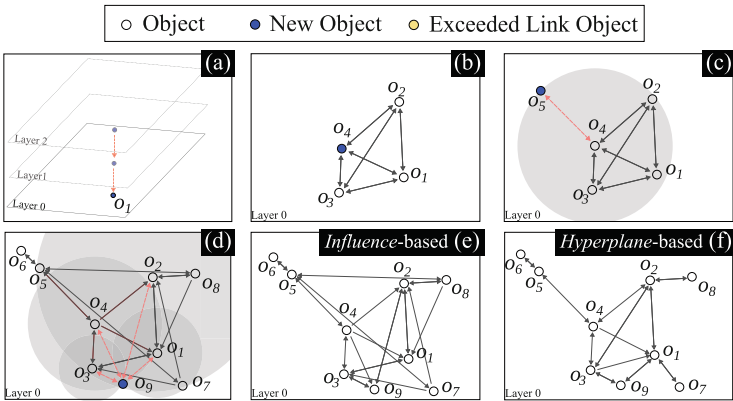


Fig. 1. The proposed *Influence*-based construction of HNSW ($M = 4$)

d HNSW) that uses *Influence* to partition the small world. Figure 1 illustrates the proposed approach for $M = 4$ and highlights the differences between HNSW and d HNSW. We maintain the HNSW skip list rationale to the last layer entry point and the fully connected graph generated for the first M objects. Thereafter, each incremental insertion creates a link to its first neighbor. We use this link to create an *Influence* region as an open ball centered at the first neighbor – Fig. 1(c). The elements covered by that open ball are not considered for creating new connections as they are *Influenced* by the first neighbor. This set of steps repeats for the next nearest neighbor (creating a new coverage ball, dismissing the *Influenced* elements) until M links are created or no valid candidates remain – Fig. 1(d). Whenever an object has more than M connections after one insert its links are refactored. This *Influence*-based strategy also creates long edges as in the hyperplane construction, but it holds closer connections together because *Influence*-driven coverage radii increase smoothly with the selection of neighbors. Such behavior is suitable for spaces where slowly exiting dense areas may yield higher Recall than quickly traversing a long edge.

A New HNSW-based Algorithm for kN_dN . We extended HNSW for *Influence*-based space partitioning and kN_dN queries. Algorithm 1 outlines the last-layer search in d HNSW with an example in Fig. 2. It relies on two priority queues (candidates \mathcal{C} , the result set \mathcal{K}) and a bitmap structure that flags evaluated objects (\mathcal{V}). The result set always includes the entry point because it is the closest neighbor and defines the first excluded *Influence* region. Thus, the candidates’ list is constructed by traversing the link set of every object included in the result set. Overall, Algorithm 1 traverses at most k paths (Line 5), each requiring at most M distance calculations (Line 8), and performs at most $M \cdot \sum_{r=1}^k r$ comparisons by *Influence* (Line 10), bounding the complexity of distance calculation to $O(kM((3+k)/2))$. Figure 1 shows a search example for object o_q and $k = 3$. The entry point is o_4 , and its set of linked elements includes o_3 that tops the list of linked neighbors.

```

px-----
dHNSW  $kN_dN$  search(Query object  $o_q$ , #neighbors  $k$ , entry point  $o_p$ );
px-----
1  $\mathcal{C} \leftarrow \{o_p\}$ ; /* Priority queue for Top-Candidates */
2  $\mathcal{K} \leftarrow \emptyset$ ; /* Priority queue for approximate Top-Neighbors */
3  $\mathcal{V} \leftarrow \{o_p\}$ ; /* Set of distinct examined/visited objects */
4  $\mathcal{L} \leftarrow \emptyset$ ; /* Auxiliary priority queue of linked neighbors in HNSW */
5 while  $\mathcal{C} \neq \emptyset \wedge |\mathcal{K}| < k - 1$  do
6    $o_i \leftarrow \mathcal{C}.\text{pop}()$ ;
7    $\mathcal{K} \leftarrow \mathcal{K} \cup \{o_i\}$ ;
8    $\mathcal{L} \leftarrow \text{linkedNeighbors}(\langle o_i, \delta(o_q, o_i) \rangle)$ ;
9   for  $o_j \in \mathcal{L}$  do
10    if  $o_j \notin \mathcal{V} \wedge o_j \notin \ddot{I}_{o_r, o_q}, \forall o_r \in \mathcal{K}$  then  $\mathcal{C} \leftarrow \mathcal{C} \cup \{o_j\}$ ;
11     $\mathcal{V} \leftarrow \mathcal{V} \cup \{o_j\}$ ;
12 return  $\mathcal{K} \cup \{o_i, o_i \leftarrow \mathcal{C}.\text{pop}()\}$ ;
px-----

```

Algorithm 1: The kN_dN searching algorithm for d HNSW.

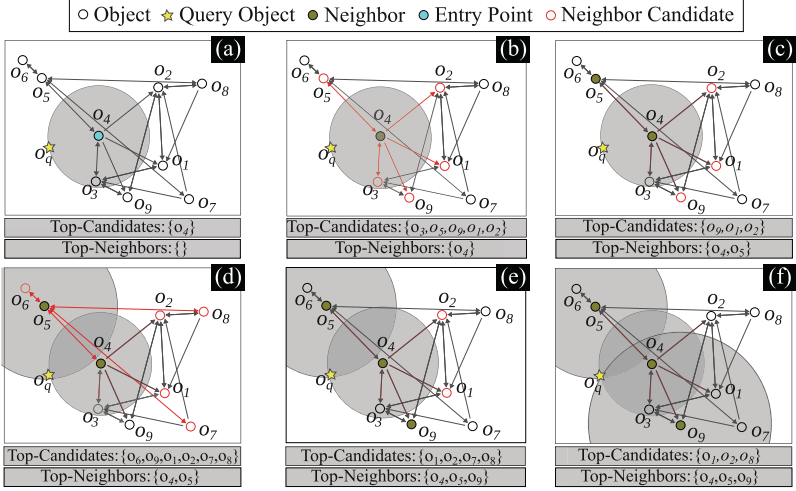


Fig. 2. Example of kN_dN query on $dHNSW$ ($k = 3$).

Implementing $dHNSW$ and kN_dN Non ANN-Benchmarks. Both $dHNSW$ and Algorithm 1 were implemented over library `nsmlib` integrated with `ANN-Benchmarks`. While `HNSW` and `dHNSW` parameters were defined in the `ANN-Benchmarks` YAML settings, the benchmark files for exact neighbors had to be replaced because of the kN_dN algorithm. Accordingly, we coupled a sequential scan to calculate the exact kN_dN solution as another `ANN-Benchmarks` Recall-driven baseline algorithm.

4 Empirical Evaluation

We used an 11-node Linux cluster with 48 AMD Opteron 2.2 GHz processors, 94 GB RAM, and 1TB SATA disk to deploy `ANN-Benchmarks`, running `HNSW` and `dHNSW` in containers. Four datasets (`FASHION-MNIST`, `GLOVE`, `MNIST`, `SIFT`) were chosen for evaluation. The following experiments compare the performances of `HNSW` and `dHNSW` in terms of quality ($\text{Recall} = (k - \sum_{o_j \in \mathcal{K}, o_i \in \mathcal{R}} |\delta(o_q, o_j) - \delta(o_q, o_i)|) / \max\{\delta(o_q, o_j), \delta(o_q, o_i)\} / k$) and throughput (queries per second – qps). The qps measure was computed as the average of five executions.

Figure 3 details the comparison between the `HNSW` and `dHNSW` regarding the execution of kN_dN queries on `ANN-Benchmarks`. Each point in the plot was juxtaposed with the exact result produced by the sequential scan, according to the parameter M employed in the graph construction. We limited $M < k$ in every evaluation and examined the neighborhood $k = \{10, 15, 20, 25\}$ with M ranging in $\{5, 10, 15, 20\}$. The y -axis represents the Recall value, while the x -axis measures the average number of qps . Figure 3 shows the outputs for the representative setup $k = 25$. Results indicate that `dHNSW` outperformed `HNSW` (up to 3% in Recall) and also delivered more qps for Recalls below 0.9 in the `MNIST`

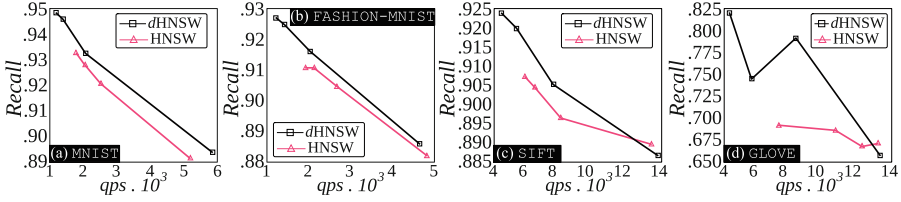


Fig. 3. Comparison between d HNSW and HNSW ($k = 20$, five executions).

dataset. The **SIFT** dataset showed similar results, with d HNSW outperforming HNSW in terms of Recall (up to 2%). Regarding **GLOVE**, d HNSW expressively outperformed HNSW (up to 27%) with a higher throughput. In the case of **FASHION-MNIST**, d HNSW also provided better Recall (up to 3%) than HNSW, the *Influence*-based strategy being slightly slower than HNSW.

5 Conclusions


In this study, we extended HNSW as d HNSW by using a *Influence*-based partitioning and proposed a new kN_dN algorithm. A benchmark evaluation indicates our approach outperformed HNSW regarding Recall with a comparable *qps*.

References

1. Aumüller, M., Bernhardsson, E., Faithfull, A.: Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Info. Sys.* **87**, 101374 (2020)
2. Drosou, M., et al.: Diversity in big data: A review. *Big Data* **5**, 73–84 (2017)
3. Fu, C., Xiang, C., Wang, C., Cai, D.: Fast approximate nearest neighbor search with the navigating spreading-out graph. *PVLDB* **12**(5) (2019)
4. Jasbick, D., et al.: Pushing diversity into higher dimensions: The LID effect on diversified similarity searching. *Info. Sys.* **114**, 102166 (2023)
5. Kucuktunc, O., Ferhatosmanoglu, H.: λ -diverse nearest neighbors browsing for multidimensional data. *TKDE* **25**(3), 481–493 (2013)
6. Malkov, Y., Yashunin, D.: Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *TPAMI* **PP** (03 2016)
7. Peng, Z., Zhang, M., Li, K., Jin, R., Ren, B.: Speed-ann: Low-latency and high-accuracy nearest neighbor search via intra-query parallelism (2022)
8. Volnyansky, I., Pestov, V.: Curse of dimensionality in pivot based indexes. In: *SISAP*. pp. 39–46. *IEEE* (2009)
9. Wang, M., et al.: A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. *PVLDB* **14**(11), 1964–1978 (2021)



An Efficient Indexing Method for Dynamic Graph k NN

Shohei Matsugu¹, Suomi Kobayashi², and Hiroaki Shiokawa¹ 

¹ Center for Computational Sciences, University of Tsukuba, Tsukuba, Japan
matsugu@kde.cs.tsukuba.ac.jp, shiokawa@cs.tsukuba.ac.jp

² Graduate School of Science and Technology, University of Tsukuba,
Tsukuba, Japan
kobayashi@kde.cs.tsukuba.ac.jp

Abstract. k -nearest neighbor (k NN) search is a fundamental problem in graph mining. This search finds the k most relevant nodes to a given query node. The increased use of social network services and map applications due to the proliferation of mobile devices has necessitated faster searches. Although pre-constructing an index using graphs can accelerate a k NN search, existing methods struggle handling dynamic graph updates. Herein we propose an efficient index update method for dynamic graphs that utilizes a core-tree structure to efficiently update the index in response to dynamic changes in the graph. Our experimental analysis using real-world data demonstrated that the proposed method can construct indexes more efficiently than the state-of-the-art method.

Keywords: graph algorithm · k NN search · dynamic graph

1 Introduction

The k nearest neighbor (k NN) search [12, 13, 17, 21] is a fundamental graph analysis tool to understand complex networks. It finds the k nearest neighbor nodes to a user-specified query node in a given graph. k NN searches are employed in diverse applications [2, 6]. Although they only need to perform a local search near the query node, k NN searches struggle handling real-world networks.

Although k NN queries are useful in many applications, they have critical drawbacks in handling real-world networks. Specifically, they require a large computational time to answer k NN queries due to the size and density of the real-world networks [15]. Historically, traditional k NN search methods are applied to small graphs such as ego-networks and road networks [4]. These methods struggle to quickly compute k NN with 10^4 nodes [21]. Recent applications based on social networks require handling large and complex networks with 10^6 nodes [14, 19], compounding the high computational costs to find k NN nodes.

1.1 Existing Approaches and Challenges

Many studies have strived to overcome the aforementioned drawbacks. One approach category is *graph indexing methods* [1, 13, 21]. These methods construct an

index using graph partitioning, which enables the shortest path distances among several nodes to be pre-computed before answering a query. Examples of graph indexing methods are G-Tree [21] and ILBR [1]. G-Tree partitions a given graph into disjointed subgraphs using Metis [8]. Then the index is constructed from the shortest path distances among all node pairs in each subgraph. Similarly, ILBR selects several landmark nodes and constructs the index from the ALT [7] values according to the shortest path distances between a node and each landmark node. Although G-Tree and ILBR improve the computation time to answer a k NN query, they still suffer from a large indexing time as they mainly focus on handling planar graphs [3] with a low density. Their indices are efficient if a given graph is sparse. However, indices in a dense graph are not computed effectively since an exhaustive pre-computation is necessary. These methods also require high computation costs for querying k NN since the coverages of the indices are too small for non-planar graphs.

CT [9,10] is a state-of-the-art k NN search method using a *core-tree-aware (CT) index* based on the *core-tree property* [5]. The core-tree property is expressed as a graph comprised of a *core* and *trees*. The core is a small and dense subgraph, while trees are long stretched and sparse subgraphs. On the basis of the core-tree property, CT constructs a core-index and a tree-index by compiling each part of the graph. Specifically, for each tree in the graph, the tree-index stores the distances from its root node to all leaf nodes. Furthermore, core-index has the distances among the remaining non-tree nodes. Using the two indices, CT can realize efficient indexing and querying for a k NN search.

Although CT can efficiently perform a k NN search, it has serious drawbacks in real-world k NN applications. CT cannot respond to node or edge updates. However, real-world graphs are frequently updated. For example, in social networks, a new edge is linked if two users become friends. If a new edge is added, CT must reconstruct the indices from scratch since it cannot update its differences. Thus, CT fails to efficiently construct the CT-index for practical use.

1.2 Our Approaches and Contributions

Our goal is to extend CT to dynamic graphs. Although the CT-index is efficient for static graphs, the index must be reconstructed to update only a few nodes and edges. Here, we present a novel indexing method called *Dynamic CT (DCT)*. The underlying idea is to update only the indices that include changed nodes. This way the tree-index is always maintained to include only the tree structures. To update the index, the process is classified according to whether the changed node is included in the core-index or the tree-index. Consequently, DCT has the following attractive characteristics:

- **Efficiency:** DCT achieves faster updates than CT (Sect. 4). DCT can perform the difference computation up to 4,462 times faster than the reconstruction of CT.
- **Exactness:** We theoretically verify that DCT always outputs the same indices as CT while achieving indexing-time improvements. It can calculate the correct k NN search results using this index.

- **Easy to deploy:** DCT does not require new indices or pre-computation processing. It achieves index updating by quickly editing the constructed CT-index.

DCT is the first solution to realize k NN searches assuming graph updates. Our extension not only increases the utility of a graph query using a k NN search but also enhances the application scope of k NN searches.

2 Preliminary

We formulate the k NN querying problem in Sect. 2.1. Then we briefly explain CT, the state-of-the-art k NN search method, in Sect. 2.2. Due to space limitations proofs of lemmas and theorems are omitted.

2.1 Problem Definition

Here, let $G = (V, E, W)$ be a weighted, undirected, and connected graph, where $V, E,$ and W are the sets of nodes, edges, and edge-weight values, respectively. $e(u, v) \in E$ denote that two nodes u and v are linked in G . For each edge $e(u, v) \in E$, an edge-weight value $w(u, v) \in W$ is always assigned, where $w(u, v) \in \mathbb{N}$ holds. The degree of node u is denoted as $deg(u)$.

k NN is a task to find k nearest neighbor nodes to a query node. We first define the shortest path distance as:

Definition 1 (Shortest path distance). *Let a node path $u = u_0 \rightarrow u_1 \rightarrow \dots \rightarrow u_i = v$ in G be the shortest path between the nodes $u, v \in V$. Here, the distance of this shortest path is defined as $dist(u, v) = \sum_{j=0}^{i-1} (w(u_j, u_{j+1}))$. Moreover, $dist_k(q, V)$ represents the k -th smallest distance in $\{dist(q, v) \mid v \in V\}$.*

By using Definition 1, we formulate the k NN query problem as:

Problem 1 (k NN query processing). *Given a graph $G = (V, E, W)$, a query node $q \in V$, and an integer $k \in \mathbb{N}$, the k NN query is to find a node set $V_k(q) = \{v \in V \mid dist(q, v) \leq dist_k(q, V)\}$.*

2.2 Previous Method: CT Index

We briefly present a *core-tree-aware (CT) indexing method* [9, 10], which is the state-of-the-art method solving Problem 1.

As mentioned in Sect. 1.1, real-world graphs often follow the core-tree property; the graphs can be decomposed into a core and trees [5]. Using this, CT constructs a CT index $\mathcal{I} = \langle \mathcal{T}, \mathcal{C} \rangle$, where \mathcal{T} and \mathcal{C} are the tree-index and core-index, respectively. CT first extracts trees from a graph and indexes them in \mathcal{T} . Then it stores the remaining core nodes in \mathcal{C} . The tree-index \mathcal{T} and the core-index \mathcal{C} are defined as follows:

Definition 2 (Tree-index \mathcal{T}). Let T_1, T_2, \dots be the trees in G and r_i be the root node of T_i . Then we denote D_i as a set of distances between r_i and each node $v \in T_i$, i.e., $D_i = \bigcup_{v \in T_i} \{dist(r_i, v)\}$. We define the tree-index as $\mathcal{T} = (\mathbb{T}, \mathbb{D})$, where \mathbb{T} and \mathbb{D} represent the sets of T_i and D_i , respectively, i.e., $\mathbb{T} = \{T_1, T_2, \dots\}$ and $\mathbb{D} = \{D_1, D_2, \dots\}$.

Definition 3 (Core-index \mathcal{C}). Let V_c be a set of core nodes that are not included in any tree-index. We define the core-index as $\mathcal{C} = (V_c, E_c, W_c)$, where $E_c = \{e(u, v) \in E \mid u, v \in V_c\}$, and $W_c = \{dist(u, v) \mid e(u, v) \in E_c\}$.

Note that the shortest path between two nodes may have multiple routes. To efficiently compute W_c , CT employs the Dijkstra algorithm to update the weight values.

For convenience, we introduce the following notation:

Definition 4 (Label function). Given a node u , the label function $f_l(u) = \text{tree}$ if $u \in \mathbb{T}$ holds, and $f_l(u) = \text{core}$ otherwise.

On the basis of the CT index, CT searches k NN nodes by applying the following lemma:

Lemma 1. Given a root node r_i in the tree T_i , and let $d_{min} = \min\{dist(q, v)\}$, where $v \in Q \cup \{v \mid e(r_i, v) \in E_c, v \notin V_k(q)\}$. If $|V_k(q)| + |T_i| \leq k$ and $d_{max}(r_i) \leq d_{min}$ hold, then $T_i \subseteq V_k(q)$ holds, where $d_{max}(v)$ is defined as the maximum shortest path distance from v to any node in T_i .

The Overview of CT: We explain the basic procedure of CT to efficiently compute k NN using the CT index. CT uses a priority queue Q to calculate the shortest path, which is similar to the Dijkstra algorithm. If q is included in the tree-index T_i in \mathcal{T} , it initially pushes r_i into Q and core-index \mathcal{C} . CT initially pushes the query node q into Q . Then it repeatedly searches k NN nodes from q using Q until it explores k nodes or reaches any root node r_i in the tree T_i . Once CT finds r_i , CT checks whether $V_k(q)$ contains all T_i . If so, CT includes T_i in $V_k(q)$ without searching T_i . Otherwise, CT explores T_i in the same way as the core node.

The main feature of CT is its application of an index construction algorithm specialized for planar graphs (e.g., traditional road networks) to more dense and diversely structured complex networks [16, 20]. However, CT is not adaptable to dynamic graphs that exist in the real world [18] as each graph update requires index reconstruction. Consequently, CT can incur a significant computation time even for minor graph changes.

3 Proposed Method: Dynamic CT

Here, we propose a novel method *Dynamic CT (DCT)* which extends CT to dynamic graphs. Here, we describe the concept of DCT and then provide details.

3.1 Ideas

We propose a method to dynamically update the CT index. In general, graph updates are represented as a collection of node or edge additions and removals. Although we focus on designing a method that accelerates the addition and removal of single edges, the proposed method does not lose generality because equivalent transformations are possible to add or remove nodes. The required processing for index updates depends on the type of transformation and the edge location (core or tree). Specifically, edge updates can change the core/tree state.

The area of this change must be limited for efficient dynamic updates. Therefore, we divide additions into four cases (Sect. 3.2) and removals into two cases (Sect. 3.3). We also theoretically calculate the update range of the graph for each case. For convenience, in the following sections, the parent node refers to the node closest to the root node on the path to any other node in the tree.

Our ideas have two advantages. First, DCT directly calculates only the index difference. By contrast, CT completely reconstructs the index for each graph update. Thus, DCT efficiently constructs the index for a graph k NN. Second, DCT always outputs the same index to CT while omitting redundant calculations. This is because DCT performs index restructuring by theoretically analyzing the area of the index affected by graph updates. Consequently, the obtained index is always the same as that constructed by CT from scratch.

3.2 Adding Nodes and Edges

We propose an algorithm that dynamically updates the CT index when nodes and edges are added to a graph. Here, only the addition of edges is considered. This is reasonable since adding a node is meaningless until an adjacent edge is added and it does not need to be distinguished from an isolated node that was already present.

The addition of an edge between nodes u and v can be classified into four cases:

Case 1. $u, v \in T_{uv}$:

Case 2. $u \in T_u$, and $v \in T_v$:

Case 3. $u \in V_c$, and $v \in T_v$:

In these three cases, we have the following property.

Lemma 2. *In Cases 1–3, $tree(s)$ are no longer a tree after adding an edge.*

Lemma 2 indicates that the tree is no longer a tree structure when an edge is added by constructing a cycle. Then we add a new cycle to the core-index \mathcal{C} . In Case 1, the cycle occurs in a tree. In Case 2, the cycle occurs across two trees. In Case 3, the cycle occurs through the tree and the core. In every case, this new cycle is added to the core-index \mathcal{C} , and the tree-index \mathcal{T} is reconstructed.

Case 4. $u, v \in V_c$:

In this case, we simply add an edge between nodes u and v since the new edge does not affect the tree-index \mathcal{T} .

Using Lemma 2, we design a novel algorithm that adds an edge to the CT-index.

Algorithm Overview: $\mathcal{I} = \langle \mathcal{T}, \mathcal{C} \rangle$ is updated using a two-fold process: linking nodes and restructuring trees. The linking step divides the cases by the location of the edge to be added. The edge is in the core or the tree. In each case, it calculates the cycle appearing through the tree-index and restructures to move it to the core-index. In the restructuring step, nodes are moved from a tree to the core. Then the node is recursively merged into its parent node for each leaf node in the updated trees.

3.3 Removing Nodes and Edges

We also introduce an algorithm to dynamically update the CT index when nodes and edges are removed from the given graph. Here, we consider only the removal of edges. Similar to adding edges, removal of a node is equivalent to the removal of all edges linked to it.

The removal of an edge between nodes u and v can be classified into two cases:

Case 1. $u, v \in V_c$: For u , if $\text{deg}(u) = 1$ holds after the removal, u and only its adjacent node w become part of the tree. In this case, u must be moved into \mathcal{T} . Thus, DCT performs restructuring from u as a leaf node. By performing the same process for v , \mathcal{T} can have the new tree that has become a tree due to the removal of the edge. By performing the same process for v , \mathcal{T} can become a new tree due to edge removal.

Case 2. $u, v \in T_{uv}$: As mentioned in Sect. 2, we assumed the graph is connected. Thus, we remove nodes that are no longer connected to the core from the graph.

Algorithm Overview: To update $\mathcal{I} = \langle \mathcal{T}, \mathcal{C} \rangle$ in DCT, the edge is initially removed. Then the cases are divided by the location of the edge to be removed. The edge is either in the core or the tree. For an edge in the core, DCT restructures if $\text{deg}(u) = 1$ or $\text{deg}(v) = 1$ holds. For an edge in a tree, DCT removes a disconnected path in the leaf side.

3.4 Complexity Analysis

Finally, we discuss the time complexity of DCT.

Theorem 1. *Updating $\mathcal{I} = \langle \mathcal{T}, \mathcal{C} \rangle$ after adding an edge incurs $O(|\bar{T}|)$ time on average, where $|\bar{T}|$ represents the average size of trees in \mathcal{T} .*

Theorem 2. *Updating $\mathcal{I} = \langle \mathcal{T}, \mathcal{C} \rangle$ after removing an edge incurs $O(\max(|V_c|, |\bar{T}|))$ time on average.*

From Theorems 1 and 2, DCT can efficiently update the index for graph changes. According to [9, 10], the index construction of CT requires $O(|E| \log |V|)$. Since $|V| > |V_c|$ and $|V| > |\bar{T}|$ hold in general, our proposed method significantly improves the computational complexity required for reconstruction compared to the state-of-the-art method.

4 Experimental Evaluation

We experimentally evaluated the efficiency of DCT compared to CT [10].

Datasets: We tested five real-world social networks [11] used in previous works [10]¹. Table 1 shows the size of the datasets.

Experimental Setup: We set $k = 0.01|V|$ as default. All experiments were conducted on a Linux server with Intel Xeon CPU 2.60 GHz and 128 GiB RAM. All algorithms were implemented in C++ using “-O2” option. We compared the running time for 100 random edge additions and removals each.

Table 1. Statistics of real-world datasets.

| | $ V $ | $ E $ | average degree |
|----|-----------|------------|----------------|
| TV | 3,892 | 17,262 | 4.4 |
| GV | 7,057 | 89,455 | 12.7 |
| NS | 27,917 | 206,259 | 7.3 |
| AT | 50,515 | 819,306 | 16.2 |
| SP | 1,632,803 | 22,301,964 | 13.7 |

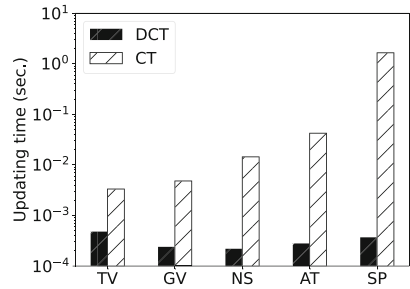


Fig. 1. Efficiency for updating edges.

4.1 Efficiency for Updating

Figure 1 shows the indexing times to add and remove edges as $m = 200$. DCT achieves significantly faster indexing for graph updates by reducing the reconstruction cost using dynamic index updates. Furthermore, CT suffers from a significant reconstruction time for large graphs because computations are performed over the entire graph. By contrast, DCT only requires computations within the neighborhood of the updated subgraph. Thus, the average degree of the graph primarily affects the update time in DCT. DCT guarantees the same results as the CT algorithm but is up to 4,462 times faster than CT.

4.2 Efficiency for Adding/Removing Edges

Figures 2 and 3 plot the index update times to add/remove only edges. DCT effectively adds and removes edges. More time is required for edge addition than removal due to the difference in the affected area for dynamic indexing. Restructuring for edge removal requires merging two paths, at most, whereas that for edge addition may involve large-scale updates that include the core and its surrounding trees.

¹ All graphs are publicly available online from <http://snap.stanford.edu/data>.

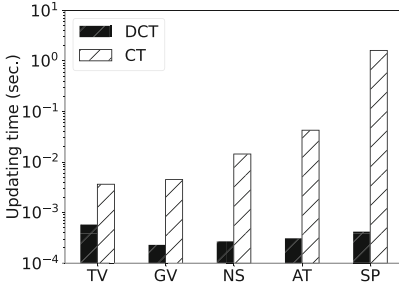


Fig. 2. Efficiency for only adding edges.

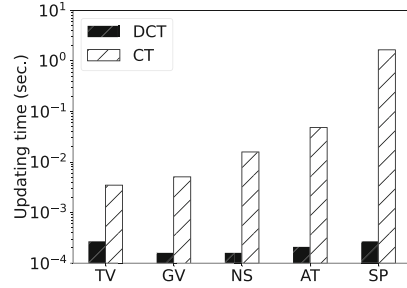


Fig. 3. Efficiency for only removing edges.

5 Conclusion

Herein a novel dynamic index update algorithm, DCT, is proposed to efficiently compute k NN searches on large-scale complex graphs. The proposed method limits the affected area of the index when the graph is updated, significantly reducing the computation time to reconstruct indexes. In our experiments, the proposed method outperforms the state-of-the-art method by up to four orders of magnitude in terms of processing times for index construction and graph k NN searches. Hence, the proposed method, which considers the core-tree characteristics, effectively reduces the cost of a k NN search on real-world dynamic graphs.

Acknowledgement. This work was partly supported by JSPS KAKENHI Grant Numbers 22K17894, 22J10972, 22KJ0398, JST PRESTO Grant Number JPMJPR2033, and JST AIP Acceleration Research JPMJCR23U2.

References












1. Abeywickrama, T., Cheema, M.A.: Efficient landmark-based candidate generation for k NN queries on road networks. In: Candan, S., Chen, L., Pedersen, T.B., Chang, L., Hua, W. (eds.) DASFAA 2017. LNCS, vol. 10178, pp. 425–440. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-55699-4_26
2. Alom, Z., Carminati, B., Ferrari, E.: Detecting spam accounts on Twitter. In: 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 1191–1198. IEEE (2018)
3. Barthélemy, M.: Morphogenesis of Spatial Networks. Springer, Cham (2018). <https://doi.org/10.1007/978-3-319-20565-6>
4. Bast, H., Funke, S., Matijević, D.: Ultrafast shortest-path queries via transit nodes. In: The Shortest Path Problem, Proceedings of a DIMACS Workshop, Piscataway, New Jersey, USA, 13–14 November 2006, vol. 74, pp. 175–192. DIMACS/AMS (2006)
5. Benson, A., Kleinberg, J.: Link prediction in networks with core-fringe data. In: The World Wide Web Conference, pp. 94–104 (2019)

6. Chen, Z., Li, P., Xiao, J., Nie, L., Liu, Y.: An order dispatch system based on reinforcement learning for ride sharing services. In: 2020 IEEE 22nd International Conference on High Performance Computing and Communications, pp. 758–763 (2020)
7. Goldberg, A.V., Harrelson, C.: Computing the shortest path: a search meets graph theory. In: SODA, vol. 5, pp. 156–165 (2005)
8. Karypis, G., Kumar, V.: Analysis of multilevel graph partitioning. In: Proceedings Supercomputing '95, San Diego, CA, USA, 4–8 December 1995, p. 29. ACM (1995)
9. Kobayashi, S., Matsugu, S., Shiokawa, H.: Indexing complex networks for fast attributed k NN queries. *Soc. Netw. Anal. Min.* **12**(1), 82 (2022)
10. Kobayashi, S., Matsugu, S., Shiokawa, H.: Fast indexing algorithm for efficient k NN queries on complex networks. In: Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 343–347. ASONAM '21, Association for Computing Machinery, New York, NY, USA (2022)
11. Leskovec, J., Krevl, A.: SNAP Datasets: Stanford Large Network Dataset Collection (2014). <http://snap.stanford.edu/data>
12. Li, R.H., Qin, L., Yu, J.X., Mao, R.: Influential community search in large networks. *Proc. VLDB Endow.* **8**(5), 509–520 (2015)
13. Li, Z., Chen, L., Wang, Y.: G*-tree: an efficient spatial index on road networks. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE), pp. 268–279. IEEE (2019)
14. Matsugu, S., Fujiwara, Y., Shiokawa, H.: Uncovering the largest community in social networks at scale. In: Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI2023), pp. 2251–2260 (2023)
15. Matsugu, S., Shiokawa, H., Kitagawa, H.: Fast and accurate community search algorithm for attributed graphs. In: Hartmann, S., Küng, J., Kotsis, G., Tjoa, A.M., Khalil, I. (eds.) Database and Expert Systems Applications. DEXA 2020. LNCS, vol. 12391, pp. 233–249. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-59003-1_16
16. Onizuka, M., Fujimori, T., Shiokawa, H.: Graph partitioning for distributed graph processing. *Data Sci. Eng.* **2**(1), 94–105 (2017)
17. Samet, H., Sankaranarayanan, J., Alborzi, H.: Scalable network distance browsing in spatial databases. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 43–54 (2008)
18. Shiokawa, H.: Scalable affinity propagation for massive datasets. In: Proceedings of the 30th AAAI Conference on Artificial Intelligence, vol. 35, no. 11, pp. 9639–9646 (2021)
19. Shiokawa, H., Amagasa, T., Kitagawa, H.: Scaling fine-grained modularity clustering for massive graphs. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence, pp. 4597–4604. IJCAI'19 (2019)
20. Shiokawa, H., Fujiwara, Y., Onizuka, M.: Scan++: efficient algorithm for finding clusters, hubs and outliers on large-scale graphs. *Proc. VLDB Endow.* **8**(11), 1178–1189 (2015)
21. Zhong, R., Li, G., Tan, K.L., Zhou, L., Gong, Z.: G-Tree: an efficient and scalable index for spatial search on road networks. *IEEE Trans. Knowl. Data Eng.* **27**(8), 2175–2189 (2015)

Database Management and Query Optimization



Improving the Accuracy of Text-to-SQL Tools Based on Large Language Models for Real-World Relational Databases

Gustavo M. C. Coelho¹ , Eduardo R. S. Nascimento¹ ,
Yenier T. Izquierdo¹ , Grettel M. García¹ , Lucas Feijó¹ ,
Melissa Lemos¹ , Robinson L. S. Garcia² , Aiko R. de Oliveira^{1,3} ,
João P. Pinheiro³ , and Marco A. Casanova^{1,3}  

¹ Instituto Tecgraf, PUC-Rio, Rio de Janeiro, RJ 22451-900, Brazil
{gustavocoelho,rogerrsn,ytorres,ggarcia,lucasfeijo,
melissa}@tecgraf.puc-rio.br

² Petrobras, Rio de Janeiro, RJ 20031-912, Brazil
robinson.garcia@petrobras.com.br

³ Departamento de Informática, PUC-Rio, Rio de Janeiro, RJ 22451-900, Brazil
{aramalho,jpinheiro,casanova}@inf.puc-rio.br

Abstract. Real-world relational databases (RW-RDB) have large, complex schemas often expressed in terms alien to end-users. This scenario is challenging to LLM-based text-to-SQL tools, that is, tools that translate Natural Language (NL) sentences into SQL queries using a Large Language Model (LLM). Indeed, their accuracy on RW-RDBs is considerably less than that reported for well-known synthetic benchmarks. This paper then introduces a technique to improve the accuracy of LLM-based text-to-SQL tools on RW-RDBs using Retrieval-Augmented Generation. The technique consists of two steps. Using the RW-RDB schema, the first step generates a synthetic dataset E of pairs (Q_N, Q_S) , where Q_N is an NL sentence and Q_S is the corresponding SQL translation. The core contribution of the paper is an algorithm that implements this first step. Given an input NL sentence Q_I , the second step retrieves pairs (Q_N, Q_S) from E based on the similarity of Q_I and Q_N , and prompts such pairs to the LLM to improve accuracy. To argue in favor of the proposed technique, the paper includes experiments with an RW-RDB, which is in production at an Energy company, and a well-known text-to-SQL prompt strategy. It repeats the experiments with Mondial, an openly available database with a large schema. These experiments constitute a second contribution of the paper.

Keywords: Text-to-SQL · Retrieval-Augmented Generation · RAG · GPT · Large Language Models · Relational Databases

1 Introduction

Natural language (NL) interfaces to databases (NLIDBs) allow users to access databases using questions formulated in Natural Language (NL) [1]. An efficient

way to construct an NLIDB is to adopt a *text-to-SQL tool* that translates NL sentences to SQL queries. A text-to-SQL tool is *generic* (or *cross-domain*) if it is designed to work with any database; by contrast, a tool is *database-specific* if it is constructed for a particular database.

A large number of generic text-to-SQL tools have been constructed with relative success [1, 6, 7] over well-known benchmarks [9, 14]. The leaderboards of these benchmarks indicate that the best tools are currently based on Large Language Models (LLMs) [10], that is, they are *LLM-based text-to-SQL tools*. However, when applied to real-world relational databases (RW-RDBs), the performance of such tools is significantly less than that reported on the leaderboards.

Indeed, RW-RDBs are challenging for at least four reasons:

1. The relational schema is often an inappropriate specification of the database from the point of view of the LLM – the table and column names are often different from the terms the users adopt to formulate their NL questions.
2. The database schema is often large, in the number of tables, columns per table, and foreign keys – but a large schema may not fit in the prompt area, and opens space to queries with many joins, which are difficult to synthesize.
3. The data semantics is often complex; for example, some data values may encode enumerated domains – again, the terms the users adopt to formulate their NL questions may have to be mapped to this internal semantics.
4. Metadata and data are often ambiguous, which influence the behavior of an LLM-based text-to-SQL tool, leading to unexpected results.

To address these challenges, Nascimento et al. [11] argued that the text-to-SQL task can be facilitated by providing a database specification based on *LLM-friendly views* that are close to the language of the users’ questions and that eliminate frequently used joins, and *LLM-friendly data descriptions* of the data semantics. However, the LLM-friendly data descriptions were limited in [11] to passing, in the prompt, a few tuples of each table used to process the NL question, which is a weak solution to capture data semantics and solve ambiguities.

This paper then concentrates on the problem of constructing database-specific LLM-based text-to-SQL tools for RW-RDBs, defined as: “Given an RW-RDB D , construct an LLM-based text-to-SQL tool such that, given any NL question on D , the tool translates the NL question into an equivalent SQL query on D ”.

The paper introduces a RAG-based technique, that is, a technique based on Retrieval-Augmented Generation [8], that provides a robust strategy to construct database-specific text-to-SQL tools for RW-RDBs, especially when it comes to conveying the data semantics of the RW-RDB to the LLM. The RAG-based technique consists of two steps. The first step generates a *synthetic dataset* E of pairs (Q_N, Q_S) , where Q_N is an NL sentence and Q_S is the corresponding SQL translation. It is based on a technique that samples the RW-RDB and its schema and associated documentation, and calls an LLM to create Q_N from the sampled data and to translate Q_N into Q_S . Roughly, by varying how the sampling works, the pairs in E help address all four RW-RDB challenges by providing text-to-SQL

examples, including pairs that expose data semantics. Therefore, the dataset E is specific to the RW-RDB, but the algorithm is generic and applicable to any RW-RDB. Given an input NL sentence Q_I , the second step retrieves samples (Q_N, Q_S) from E based on the similarity of Q_I and Q_N , and prompts such pairs to the LLM to improve accuracy. The core contribution of the paper is an algorithm to generate a synthetic dataset E from an RW-RDB and its schema and associated documentation.

To argue in favor of the RAG-based technique, the paper includes experiments with the same RW-RDB and the same set of questions as in [11], and a text-to-SQL prompt strategy, implemented with LangChain using GPT-3.5 and GPT-4, which includes the RAG-based technique. The results suggest that the RAG-based technique indeed leads to improved accuracy. Since the RW-RDB is proprietary, the paper repeats the experiments with Mondial, an openly available database with a large, complex relational schema. As the Mondial schema adopts a user-friendly vocabulary, using LLM-friendly views is not required. On these challenging databases, the proposed RAG-based technique achieved an accuracy close to that obtained by the best strategies on the (much simpler) benchmark databases. These experiments constitute a second contribution of the paper.

This paper is organized as follows. Section 2 covers related work. Section 3 summarizes the RW-RDB benchmark adopted and the LLM-friendly views solution. Section 4 details the RAG-based technique. Section 5 describes the experiments with the RW-RDB. Section 6 summarizes the experiments with Mondial. Finally, Sect. 7 contains the conclusions.

2 Related Work

Text-to-SQL Datasets. The Spider – Yale Semantic Parsing and Text-to-SQL Challenge [14] defines 200 datasets, covering 138 domains, for training and testing text-to-SQL tools. For each database, Spider lists 20–50 hand-written NL questions and their SQL translations. An NL question Q_N , with an SQL translation Q_S , is classified as easy, medium, hard, and extra-hard, where the difficulty is based on the number of SQL constructs of Q_S . The set of NL questions introduced in Sect. 3.3 follows this classification.

Most databases in Spider have small schemas: the largest five databases have between 16 and 25 tables, and about half of the databases have schemas with five tables or fewer. Also, all Spider NL questions are phrased in terms used in the database schemas. These two limitations considerably simplify the text-to-SQL task. Therefore, the results reported in the Spider leaderboard are biased toward databases with small schemas and NL questions written in the schema vocabulary, which is not what one finds in real-world databases.

Spider has two interesting variations. Spider-Syn [2] is used to test how well text-to-SQL tools handle synonym substitution, and Spider-DK [3] addresses testing how well text-to-SQL tools deal with domain knowledge.

BIRD – BIg Bench for LaRge-scale Database Grounded Text-to-SQL Evaluation [9] is a large-scale cross-domain text-to-SQL benchmark in English. The

dataset contains 12,751 text-to-SQL data pairs and 95 databases with a total size of 33.4 GB across 37 domains. However, BIRD still does not have many databases with large schemas: of the 73 databases in the training dataset, only two have more than 25 tables, and, of the 11 databases used for development, the largest one has only 13 tables.

Despite the availability of these benchmarks for text-to-SQL, and inspired by them, Sect. 3 describes a benchmark tuned to the problem addressed in this paper. The benchmark consists of a relational database, whose design is based on a real-world database, three sets of LLM-friendly views, specified as proposed in [11], and a set of 100 test NL questions, that mimic those posed by real users, and their ground truth SQL translations.

Text-to-SQL Tools. The Spider Web site¹ publishes a leaderboard with the best-performing text-to-SQL tools. At the time of this writing, the top 5 tools achieved an accuracy that ranged from an impressive 85.3% to 91.2% (two of the tools are not openly documented). Four tools use GPT-4, as their names imply. The three tools that provide detailed documentation have an elaborate first prompt that tries to select the tables and columns that best matches the NL question. The first prompt is, therefore, prone to failure if the database schema induces a vocabulary which is disconnected from the NL question terms. This failure cannot be easily fixed by even more elaborate prompts that try to match the schema and the NL question vocabularies, as argued in [11].

The BIRD Web site² also publishes a leaderboard. At the time of this writing, out of the top 5 tools, two use GPT-4, one uses CodeS-15B, one CodeS-7B, and one is not documented. The sixth and seventh tools also use GPT-4, appear in the Spider leaderboard, and are well-documented.

Finally, LangChain³ is a generic framework that offers several predefined strategies to build and run SQL queries based on NL prompts. Section 5.1 uses LangChain to create a text-to-SQL prompt strategy.

Retrieval-Augmented Generation – RAG. Retrieval-Augmented Generation (RAG), introduced in [8], is a strategy to incorporate data from external sources before proceeding to the generation phase. This process ensures that the responses are grounded in retrieved evidence, thereby significantly enhancing the accuracy and relevance of the output.

There is an extensive literature on RAG. Recent references include a RAG technique for an LLM-based Text-to-SQL framework involving sample-aware prompting and a dynamic revision chain [5]. A RAG technique is used in [13] to retrieve the table and column descriptions to ensure that the NL question is related to the right tables and columns. A recent survey can be found in [4], encompassing the *Naive RAG*, the *Advanced RAG*, and the *Modular RAG*.

The core contribution of the paper is an algorithm that, given an RW-RDB D_R and its schema D_S and associated documentation D_{doc} , generates a synthetic

¹ <https://yale-lily.github.io/spider>.

² <https://bird-bench.github.io>.

³ <https://python.langchain.com>.

dataset E of pairs (Q_N, Q_S) , where Q_N is an NL question and Q_S is its SQL translation, using D_R , D_S and D_{doc} . The synthetic dataset E is then used in an RAG technique to improve the accuracy of a text-to-SQL strategy on D_R .

3 A Real-World Benchmark for the Text-to-SQL Task

This section describes a benchmark to help investigate the text-to-SQL task over an RW-RDB. It should be stressed that this benchmark was designed exclusively for testing text-to-SQL tools; it was not meant for training such tools.

3.1 The Real-World Relational Database

The RW-RDB is proprietary and stores data related to the integrity management of an energy company’s industrial assets. The relational schema contains 27 relational tables with, in total, 585 columns and 30 foreign keys (some multi-column); the largest table has 81 columns.

Table and column names in the relational schema do not follow a specific vocabulary. This scenario implies that end-users have difficulty understanding the semantics of the stored data and must turn to database specialists, even if they have access to the database documentation.

But there is a second, more challenging problem: some column values are not end-user-friendly. Indeed, some column values, or combinations of column values, hide semantic information that does not directly correspond to end-user terms. To overcome this situation, database experts often create SQL functions that contain the logic to represent the semantics hidden in the column values. Still, end-users adopt their vocabulary to refer to these data values, such as “overdue order” which translates to “`Maintenance_Order.Status = 1`”; this translation is in the database documentation, but it is not readily visible.

3.2 The Sets of Views

To avoid the effect of the internal naming convention of the RW-RDB, the benchmark introduces three sets of LLM-friendly views of increasing complexity:

- *Conceptual schema views*: a set of views that define a one-to-one mapping of the relational schema to end users’ terms; the views basically rename tables and columns.
- *Partially extended views*: a set of views that extend the conceptual schema views with new columns that predefine joins that follow foreign keys, as well as other selected columns.
- *Fully extended views*: a set of views such that each view combines several conceptual schema views; the set may optionally include some conceptual schema views.

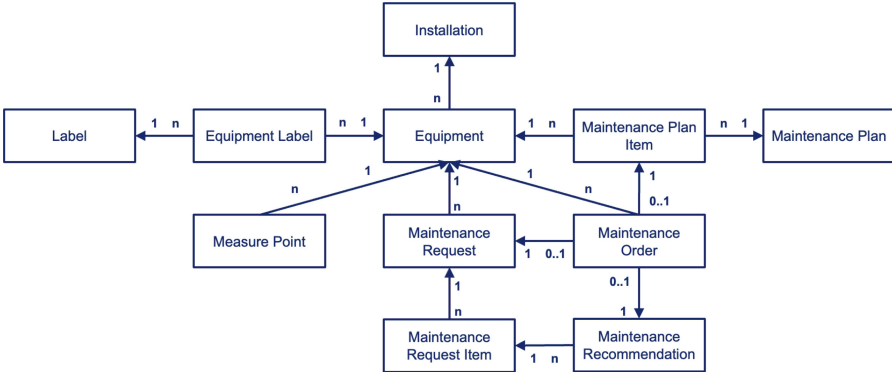


Fig. 1. The referential dependency diagram of a simplified version of the RW-RDB.

Table 1. A sample of the NL questions and their translations.

| ID | NL Question | Ground Truth SQL Query Table | Question Type |
|----|--|---|---------------|
| 1 | Which IBX15 installation recommendations are expired? | <pre> SELECT id FROM vw_maintenance_recomentation WHERE expired = 'true' AND installation_name = 'IBX-15' </pre> | medium |
| 25 | Show all orders with type MO05 | <pre> SELECT id FROM vw_maintenance_order WHERE type = 'MO05' </pre> | simple |
| 47 | Which IBX-67 installation label has the most approved maintenance requests with a priority greater than 6? | <pre> SELECT el.label_id FROM vw_maintenance_request mr JOIN vw_equipment_label el ON mr.equipment_id = el.equipment_id WHERE mr.installation_name = 'IBX-67' AND mr.situation = 'Approved' AND mr.priority > 6 GROUP BY el.label_id ORDER BY COUNT(mr.id) DESC FETCH FIRST 1 ROWS ONLY </pre> | complex |

Figure 1 shows the referential dependencies diagram of a much-simplified version of the conceptual schema views, where an arrow represents a foreign key and points to the referenced table, as usual.

The set of *partially extended views* was defined by including the non-primary key columns of the view `Installation` into the views `Equipment`, `Maintenance_Order`, `Maintenance_Request`, `Maintenance_Recommendation`, and `Maintenance_Plan_Item`. The following statement shows the SQL code that creates the partially extended view `PE_Equipment` by combining the views `Installation` and `Equipment`:

```

CREATE VIEW pe_equipment AS
SELECT inst.name AS installation_name, inst.asset,
       inst.main_hub, inst.business_unit, equip.*
FROM equipment equip JOIN installation inst
  ON inst.id = equip.installation_id
  
```

where the `Installation` view has columns `id` (*primary key*), `name`, `asset`, `main_hub`, and `business_unit`, and the `installation_id` column of the `Equipment` view is a foreign key to the `id` column of `Installation`.

Finally, the set of fully extended views was defined following a similar strategy but combining two or more conceptual schema views. For instance, view `FE_Installation_Equipment_Maintenance_Request` combines views `Installation`, `Equipment`, and `Maintenance Request`.

3.3 The Test Questions and Their Ground Truth SQL Translations

The benchmark contains a set of 100 NL questions, $L = \{L_1, \dots, L_{100}\}$, that consider the terms and questions experts use when requesting information related to the maintenance and integrity processes.

The ground truth SQL queries, $G = \{G_1, \dots, G_{100}\}$, were manually defined over the conceptual schema views so that the execution of G_i returns the expected answer to the NL question L_i . The use of the conceptual schema views facilitated this manual task, since these views use a vocabulary close to that of the NL questions.

An NL question L_i is classified into *simple*, *medium*, and *complex*, based on the complexity of its ground truth SQL query G_i , as in the Spider benchmark (extra-hard questions were not considered). The set L contains 33 simple, 33 medium, and 34 complex questions.

Note that the NL question classification is anchored on the conceptual schema views. But, since these views map one-to-one to the tables of the relational schema, a classification anchored on the relational schema would remain the same. The classification is maintained for the other sets of views, even knowing that the definition of these other sets of views might simplify the translation of some NL questions (which was one of the reasons for considering these sets of views in the first place).

Table 1 shows some NL questions and their ground truth SQL translations.

4 The Proposed RAG-Based Technique

4.1 Generation of the Synthetic Dataset

A *synthetic dataset* may provide SQL examples illustrating how the database schema is structured, how the user’s language maps to the database schema, and how NL language constructions map to data values. This section outlines a procedure to generate a synthetic dataset with examples of all these three types, by exploring the database, its schema, and associated documentation.

Algorithm 1 shows a much simplified pseudo-code of the core procedure, which generates a pair (Q_N, Q_S) , where Q_N is a NL question and Q_S is the corresponding SQL query. Very briefly, the core procedure goes as follows:

Algorithm 1: GenerateExample

Input: the number n of columns to select, the database D_R , the database schema D_S , and the database documentation D_{doc} , if available.

Output: a pair (Q_N, Q_S) where Q_N is an NL question and Q_S is the corresponding SQL query.

```

1 Function GenerateExample( $n, D_R, D_S, D_{doc}$ ):
2    $Q_A \leftarrow \text{SelectColumns}(n, D_S)$ ;
3    $Q_K \leftarrow \text{CreateNLQuestion}(Q_A, D_R, D_S, D_{doc})$ ;
4    $Q_S \leftarrow \text{CompileSQLQuery}(Q_K, D_S)$ ;
5    $Q_N \leftarrow \text{ImproveNLQuestion}(Q_K, D_R, D_S, D_{doc})$ ;
6   return  $(Q_N, Q_S)$ ;

```

- Step 1 (on Line 2) selects a set Q_A of n pairs of table/column names from the database tables. The selection process employs a weighted random distribution, which reflects the likelihood of each column of a table being chosen by an average user.
- Step 2 (on Line 3) creates an NL question Q_K from Q_A by prompting GPT-3.5-turbo with the following information: the column/table pairs selected in Step 1, sample values of each column/table, and a simplified Data Definition Language (DDL) statement encompassing only the columns and tables involved, including any join tables. In addition, the type of restriction to be incorporated into the NL question depends on the nature of the data. For instance, for numerical columns, restrictions may involve operations such as summation, averaging, or finding the maximum value. Similarly, requests might include grouped aggregations for categorical columns, among other possibilities. Lastly, the prompt includes instructions on how to formulate the NL question by using the database vocabulary without altering the column and table names. This is essential for the next step.
- Step 3 (on Line 4) calls GPT-4 to translate Q_K into an SQL query that responds to the NL question by providing the simplified DDL statement in the same manner as in Step 2. It is worth noting that, since Q_K is written using the database vocabulary and since the DDL statement describes only the necessary tables and columns, this translation task is relatively simple.
- Finally, Step 4 (on Line 5) calls GPT-3.5-turbo to translate Q_K into an improved NL question Q_N , using the database documentation D_{doc} , which includes the Description of each column and table along with synonyms. During this step, GPT-3.5-turbo is instructed to rephrase the NL question by translating from the database to the user’s vocabulary, preserving the original NL question intent.

By looping Algorithm 1 over different combinations of table/column name samples, one can generate a reasonably large dataset, containing thousands of instances of NL questions and their corresponding SQL queries.

When $n = 1$, Algorithm 1 samples just one column/table pair from the database schema. This option is interesting for capturing data value semantics, as the following example illustrates:

- Suppose Step 1 selects column `Classification` of table `Maintenance_Plan`.
- Since `Classification` is a categorical column, suppose Step 2 decides to create the filter `Classification = 1`, based on the samples provided in the prompt. The NL question will then be $Q_K = \text{“List all instances on the Maintenance table which have Classification equal to 1”}$.
- Step 3 creates the SQL query Q_S

```
SELECT * FROM Maintenance_Plan WHERE Classification = 1;
```

- Step 4 observes in the database documentation that users adopt *Critical Plans* to refer to plans such that `Classification = 1`. Therefore, Step 4 generates the improved NL question $Q_N = \text{“List all critical plans”}$.

When $n > 1$, Algorithm 1 samples two or more column/table pairs from the database schema, which forces Step 3 to generate SQL queries with one or more joins, if the sampled column/table pairs are from different tables. For example, consider a sample with two column/table pairs (that is, $n = 2$):

- Suppose Step 1 selects column `Description` of table `Maintenance_Request` and column `Code_Name` of table `Installation`.
- Based again on the samples provided and the nature of the columns, suppose that Step 2 generates the following question: $Q_K = \text{“List the Code_Name instances from the table Installation that are equal to XPTO associated with Description containing the word “paint” from table Installation.}.$
- Step 3 creates the SQL query Q_S

```
SELECT M.ID
FROM Maintenance_Request M
JOIN Equipment E ON E.ID = M.Equipment_ID
JOIN Installation I
ON I.ID = E.Installation_ID
WHERE I.Code_Name = 'XPTO' AND
LOWER(M.Description) LIKE '%paint%';
```

- Step 4 then improves the readability of Q_K , generating the NL question $Q_N = \text{“List all paint maintenance requests for installation XPTO”}$.

Note that the SQL query Q_S contains two joins to navigate from table `Maintenance_Request` to table `Installation`, via table `Equipment`, as depicted in Fig. 1. That is, relating column `Description` of table `Maintenance_Request` and column `Code_Name` of table `Installation` is somewhat more challenging.

Finally, we observe that the implementation of the core procedure is capable of generating far more complex NL question/SQL query pairs. Without going into the details, the following example illustrates this remark:

- Initial NL question: *“How many maintenance orders from table Maintenance_Order are associated with each classification category from table Maintenance_Plan, considering only maintenance orders whose Description contains the word ‘paint’?”*

- Reformulated NL question: “How many maintenance orders are associated with each maintenance plan classification, considering only maintenance orders whose Description contains the word ‘paint’?”
- SQL query for both NL questions:

```

SELECT P.Classification,
       COUNT(M.ID) AS Measurement_Quantity
FROM Maintenance_Plan P
     JOIN Maintenance_Plan_Item I
       ON P.ID = I.Plan_ID
     JOIN Maintenance_Order M
       ON I.ID = M.Item_ID
WHERE LOWER(M.Description) LIKE '%paint%'
GROUP BY P.Classification;

```

4.2 The Proposed RAG-Based Techniques

The RAG-based techniques assume that the NL questions in the synthetic dataset have already been embedded into a vector space and indexed accordingly. The critical step, *question similarity selection*, first obtains an embedding E_I of the input NL question. Then, it retrieves from the synthetic dataset the top-k pairs whose NL question embeddings are similar to E_I , as usual.

The experiments will consider four configurations, which test two synthetic datasets combined or not with schema information.

The *single-attribute synthetic dataset* is generated by sampling just single attributes (that is, by calling Algorithm 1 always with $n = 1$), whereas the *multi-attribute synthetic dataset* is generated by sampling multiple attributes. Therefore, the single-attribute syntactic dataset will contain only pairs (Q_N, Q_S) , where Q_N is a *simple NL question* and Q_S is a SQL query over a single table, with no join clauses, and a WHERE clause with one filter over a single column.

Now, *RAG with no schema information* uses RAG to retrieve examples from the synthetic dataset which are similar to the input NL question, and prompts the LLM only with the retrieved examples. These experiments test whether the synthetic dataset is sufficient to convey all the information about the database the LLM requires for the text-to-SQL task.

By contrast, *RAG with schema information* uses RAG to retrieve examples from the synthetic dataset and prompts the LLM with the retrieved examples and the database schema. These experiments test whether the RAG-based technique adds information not conveyed by the schema, thereby leading to a better text-to-SQL prompt strategy.

5 Experiments with an RW-RDB

5.1 Experimental Setup

Benchmark. The experiments used the RW-RDB benchmark defined in Sect. 3, with the partially extended views, which achieved the best performance in [11].

Performance Indicator. The experiments used the *accuracy* of a given text-to-SQL strategy over the benchmark, defined as the number of correct predicted SQL queries divided by the total number of SQL queries, as usual.

The experiments used an automated procedure to compare the *predicted* and the *ground truth* SQL queries, entirely based on column and table values, and not just column and table names. Therefore, a text-to-SQL tool may generate SQL queries over the relational schema or any set of views, and the resulting SQL queries may be compared with the ground truth SQL queries based on the results returned. The results of the automated procedure were manually checked to eliminate false positives and false negatives.

Setup Configurations. The experiments were based on a text-to-SQL implementation using LangChain SQLQueryChain which automatically extracts metadata from the database, creates a prompt with the metadata and passes it to the LLM. This chain greatly simplifies creating prompts to access databases through views since it passes a view specification as if it were a table specification. This strategy was adopted because it proved inexpensive and had an accuracy compared with much more complex text-to-SQL tools [12].

The multi-attribute synthetic dataset had 46,215 pairs created on top of the partially extended views, whereas the single-attribute synthetic dataset had 24,861 pairs. The usual cosine similarity function was adopted to compare the user’s NL question and the NL questions in the dataset.

The experiments tested several configurations, involving different models, strategies, and sample sizes, detailed in the next section to avoid repetition.

5.2 Results

Table 2 shows the results obtained with the different setup configurations. Column **Model** indicates the models used; column **#Samples**, the number of samples retrieved from the synthetic dataset; columns under **#Correct Predicted Queries**, the number of simple, medium, or complex SQL queries correctly synthesized; columns under **Accuracy**, the accuracy results for simple, medium, or complex SQL queries, recalling that the benchmark had 33 simple, 33 medium, and 34 complex NL questions. The lines of Table 2 should be read as follows:

- Lines 1 and 2 show the results when the LLM (GPT-3.5 or GPT-4) is prompted with the relation tables. These are the baselines.
- Lines 3 and 4 show the results using the RAG technique, with the multi-attribute synthetic dataset, and without schema information. The experiments used GPT-3.5-turbo and GPT-4, and retrieved the top 15 most similar pairs from the synthetic dataset.
- Lines 5 and 6 show the results, obtained in [11], using the LLM-friendly partially extended views, without the RAG-based technique.
- Lines 7 and 8 show the results using the RAG-based technique, with the multi-attribute synthetic dataset, combined with the LLM-friendly partially extended views. The experiments used GPT-3.5-turbo-16K and GPT-4-32K,

since the prompt turned out to be large, and retrieved the top-8 most similar pairs from the dataset, as well as three rows for each table as samples.

- Lines 9 and 10 show the results using the same configuration as in Lines 7 and 8, except that the RAG-based technique used the single-attribute synthetic dataset.

The rest of this section discusses the results of the setup configurations that use the RAG-based technique in more detail.

Table 2. Results for the different setup configurations – RW-RDB.

| #Line | Model (all with SQLQueryChain) | #Samples | #Correct Predicted Queries | | | | Accuracy | | | |
|--|--------------------------------|----------|----------------------------|--------|---------|-------|----------|--------|---------|-------------|
| | | | Simple | Medium | Complex | Total | Simple | Medium | Complex | Total |
| <i>Relational Schema</i> | | | | | | | | | | |
| 1 | GPT-3.5 | - | 24 | 8 | 4 | 36 | 0,73 | 0,24 | 0,11 | 0,36 |
| 2 | GPT-4 | - | 22 | 11 | 8 | 41 | 0,67 | 0,33 | 0,23 | 0,41 |
| <i>RAG-Based Techniques Only</i> | | | | | | | | | | |
| 3 | GPT-3.5-turbo | Top 15 | 24 | 10 | 3 | 37 | 0,73 | 0,30 | 0,09 | 0,37 |
| 4 | GPT-4 | Top 15 | 26 | 7 | 9 | 42 | 0,79 | 0,21 | 0,26 | 0,42 |
| <i>Partially Extended Conceptual Schema Views Only</i> | | | | | | | | | | |
| 5 | GPT-3.5 | - | 26 | 18 | 8 | 52 | 0,79 | 0,54 | 0,23 | 0,52 |
| 6 | GPT-4 | - | 30 | 25 | 19 | 74 | 0,91 | 0,76 | 0,56 | 0,74 |
| <i>RAG-Based Techniques with the Multi-Attribute Synthetic Dataset and the Partially Extended Conceptual Schema Views</i> | | | | | | | | | | |
| 7 | GPT-3.5-turbo-16K | Top 8 | 30 | 16 | 9 | 55 | 0,91 | 0,48 | 0,26 | 0,55 |
| 8 | GPT-4-32K | Top 8 | 31 | 26 | 19 | 76 | 0,94 | 0,79 | 0,56 | 0,76 |
| <i>RAG-Based Techniques with the Single-Attribute Synthetic Dataset and the Partially Extended Conceptual Schema Views</i> | | | | | | | | | | |
| 9 | GPT-3.5-turbo-16K | Top 8 | 30 | 20 | 11 | 61 | 0,91 | 0,61 | 0,32 | 0,61 |
| 10 | GPT-4-32K | Top 8 | 32 | 24 | 23 | 79 | 0,97 | 0,73 | 0,68 | 0,79 |

RAG-based Technique Only. Consider first the results for the configurations using SQLQueryChain over the partially extended views, with the RAG-based technique using the multi-attribute synthetic dataset, but without passing any schema information (Lines 3 and 4). GPT-3.5 and GPT-4 had relatively low overall accuracy – 37% and 42%, respectively. GPT-4 performed better on simple and complex queries, while GPT-3.5 performed better on medium queries. GPT-4 correctly answered 79% of the simple queries.

In summary, the first experiment suggested that:

- The RAG-based technique provides sufficient information for the LLM to process most of the simple NL questions.
- In more complex NL questions, the examples provided by the RAG-based technique may not include all the necessary information.
- Without knowing the database schema, the LLM may “hallucinate” by using tables and columns that do not exist, or using tables mentioned in the examples but that do not correspond to the user NL question.

Schema information is therefore necessary for generating more complex SQL queries – just the RAG-based technique was insufficient.

RAG-Based Technique with LLM-friendly Partially Extended Views. Consider now the results for the configurations using the RAG-based technique

combined with LLM-friendly partially extended views (Lines 7–10). The best result was obtained with the single-attribute synthetic dataset (lines 9 and 10). GPT-4 with the top-8 samples had the best performance with a 79% accuracy; it surpassed the previous best configuration using only the LLM-friendly partially extended views (Line 6), which achieved a 74% accuracy. For simple NL questions, this configuration achieved the best performance, with a 97% accuracy. For complex NL questions, it also achieved the best performance with a 68% accuracy; this represents a significant improvement over the previous best approach for complex NL questions, which used only the LLM-friendly partially extended views (Line 6), and achieved a 56% accuracy. However, for medium NL questions, the configuration actually had a lower accuracy than the previous best configuration for medium NL questions (Line 6), as well as lower than RAG with the multi-attribute synthetic dataset (Line 8).

In summary, the second experiment suggested that the RAG-based technique:

- Allowed an LLM to understand the “vocabulary of the database schema”. For example, in schema linking, it enabled the LLM to correctly generate SQL queries in ambiguous NL questions such as “out of date” (`out_of_date = yes`) and “canceled orders” (`order.status = ‘Canceled’`).
- Increased accuracy to 97% on simple NL questions and to 68% on complex NL questions, whereas the previous best approach reached 91% and 56%, respectively.

Thus, the second experiment suggested that the RAG-based technique with the single-attribute synthetic dataset helped achieve non-trivial improvements on previous results, obtained using only LLM-friendly partially extended views.

6 Experiments with Mondial

Benchmark. The second set of experiments used the Mondial database⁴ and 100 NL questions and their *ground truth* translations to SQL queries⁵, divided into 34 simple, 33 medium, and 33 complex questions. Mondial stores geographic data and is openly available. It has a total of 47.699 instances; the relational schema has 46 tables, with a total of 184 columns and 49 foreign keys, some of which are multi-column.

Performance Indicator. (Same as in Sect. 5.1).

Setup Configurations. The experiments tested two configurations based on SQLQueryChain, using GPT-3.5-turbo-16K and GPT-4, with RAG on a multi-attribute synthetic dataset with 60,000 pairs, created on top of the Mondial relational schema and associated documentation.

Results. Lines 1 and 2 of Table 3 repeat the best results obtained in [12] for Mondial without using RAG, taken as the baselines; Lines 3 and 4 show the results for

⁴ <https://www.dbis.informatik.uni-goettingen.de/Mondial/>.

⁵ Available on request.

the two configurations with RAG. The results corroborate the findings reported in Sect. 5.2. The RAG-based technique with GPT-4 obtained an accuracy of 86% (Line 4), surpassing the previous best result of 78% (Line 2), obtained using C3 with GPT-4. The RAG-based technique using GPT-3.5-turbo-16K achieved an accuracy of 72% (Line 3), surpassing all previous strategies tested, except for the RAG-based technique with GPT-4 and C3 with GPT-4. The use of RAG substantially improved the accuracy for medium and complex NL questions. But, for simple NL questions, the multiple similar examples confused the LLM.

Table 3. Results for the different setup configurations – Mondial.

| #Line | Strategy / Model | #Samples | #Correct Predicted Queries | | | | Accuracy | | | |
|--|--------------------------------------|----------|----------------------------|--------|---------|-------|----------|--------|---------|-------------|
| | | | Simple | Medium | Complex | Total | Simple | Medium | Complex | Total |
| Relational Schema | | | | | | | | | | |
| 1 | SQLQueryChain with GPT-3.5-turbo-16k | 3 | 25 | 22 | 13 | 60 | 0,76 | 0,67 | 0,38 | 0,60 |
| 2 | C3 with GPT-4 | - | 29 | 25 | 24 | 78 | 0,88 | 0,76 | 0,71 | 0,78 |
| RAG-Based Techniques with the Multi-Attribute Synthetic Dataset and the Relational Schema | | | | | | | | | | |
| 3 | SQLQueryChain with GPT-3.5-turbo-16K | Top 8 | 28 | 23 | 21 | 72 | 0,85 | 0,70 | 0,62 | 0,72 |
| 4 | SQLQueryChain with GPT-4 | Top 8 | 28 | 31 | 27 | 86 | 0,85 | 0,94 | 0,79 | 0,86 |

7 Conclusions

This paper argued that a RAG-based technique provides a robust strategy to construct database-specific text-to-SQL tools for RW-RDBs. The key step is based on an algorithm that samples the RW-RDB schema and the associated documentation, and calls an LLM to create NL questions and their translation to SQL from the sampled data. By varying how the sampling works, this step generates a dataset containing pairs of NL question/SQL query that give examples of how to create SQL queries with several joins over the RW-RDB adopted, as well as examples that expose the semantics of the data stored in the RW-RDB. The algorithm is generic and applicable to other RW-RDBs.

To argue in favor of the RAG-based technique, the paper included experiments with the same RW-RDB and the same set of questions as in [11], and a text-to-SQL prompt strategy, implemented with LangChain using GPT-3.5 and GPT-4, which includes the RAG-based technique. The results for the RW-RDB suggested that the RAG-based technique improved accuracy, especially for complex NL questions. The results for Mondial corroborated these findings.

In future work, we plan to expand the experiment to other RW-RDBs, and with variations of the RAG-based technique, including changing the similarity function and working with a different strategy that retrieves a diversified list of pairs, rather than just the top-n. We also plan to expand the approach to cover different user groups, with distinct and often contradictory vocabularies.

Acknowledgements. This work was partly funded by FAPERJ under grant E-26/202.818/2017; by CAPES under grants 88881.310592-2018/01, 88881.134081/2016-01, and 88882.164913/2010-01; by CNPq under grant 302303/2017-0; and by Petrobras.

References

1. Affolter, K., Stockinger, K., Bernstein, A.: A comparative survey of recent natural language interfaces for databases. *VLDB J.* **28** (2019). <https://doi.org/10.1007/s00778-019-00567-8>
2. Gan, Y., et al.: Towards robustness of text-to-sql models against synonym substitution. *CoRR* **abs/2106.01065** (2021). <https://doi.org/10.48550/arXiv.2106.01065>
3. Gan, Y., Chen, X., Purver, M.: Exploring underexplored limitations of cross-domain text-to-sql generalization. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 8926–8931, January 2021. <https://doi.org/10.18653/v1/2021.emnlp-main.702>
4. Gao, Y., et al.: Retrieval-augmented generation for large language models: a survey. *arXiv preprint* (2024). <https://doi.org/10.48550/arXiv.2312.10997>
5. Guo, C., et al.: Retrieval-augmented gpt-3.5-based text-to-sql framework with sample-aware prompting and dynamic revision chain. *arXiv preprint* (2023). <https://doi.org/10.48550/arXiv.2307.05074>
6. Katsogiannis-Meimarakis, G., Koutrika, G.: A survey on deep learning approaches for text-to-SQL. *VLDB J.* **32**(4), 905–936 (2023). <https://doi.org/10.1007/s00778-022-00776-8>
7. Kim, H., So, B.H., Han, W.S., Lee, H.: Natural language to SQL: where are we today? *Proc. VLDB Endow.* **13**(10), 1737–1750 (2020). <https://doi.org/10.14778/3401960.3401970>
8. Lewis, P., et al.: Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Larochele, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474. Curran Associates, Inc. (2020). <https://api.semanticscholar.org/CorpusID:218869575>
9. Li, J., et al.: Can LLM already serve as a database interface? A big bench for large-scale database grounded text-to-SQLs. *arXiv preprint* (2023). <https://doi.org/10.48550/arXiv.2305.03111>
10. Manning, C.D.: Human language understanding & reasoning. *Daedalus* **151**(2), 127–138 (2022). https://doi.org/10.1162/daed_a.01905
11. Nascimento, E.R., et al.: My database user is a large language model. In: *Proceedings of the 26th International Conference on Enterprise Information Systems*, vol. 1, pp. 800–806 (2024). <https://doi.org/10.5220/0012697700003690>
12. Nascimento, E.R., et al.: Text-to-SQL meets the real-world. In: *Proceedings of the 26th International Conference on Enterprise Information Systems*, vol. 1, pp. 61–72 (2024). <https://doi.org/10.5220/0012555200003690>
13. Panda, S., Gozhluklu, B.: Build a robust text-to-sql solution generating complex queries, self-correcting, and querying diverse data sources. *AWS Machine Learning Blog*, 28 February 2024
14. Yu, T., et al.: Spider: a large-scale human-labeled dataset for complex and cross-domain semantic parsing and Text-to-SQL task. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3911–3921, Oct–Nov 2018. <https://doi.org/10.18653/v1/D18-1425>



QPSEncoder: A Database Workload Encoder with Deep Learning

Jianwen Yang^{1,2}, QiuHong Zhang^{1,2}, Jin Yan^{1,2(✉)}, Zhiming Ding²,
Meiling Zhu², and Xinjie Lv³

¹ University of Chinese Academy of Sciences, Beijing, China
{yangjianwen18,zhangqiuHong22}@mailsucas.ac.cn,
yvette.yan@mailsucas.edu.cn

² Institute of Software, Chinese Academy of Sciences, Beijing, China
{zhiming,meiling}@isacs.ac.cn

³ Highgo Infrastructure Software Co., Ltd., Jinan, China
xinjie@highgo.com

Abstract. Machine learning has become a prominent approach for many database optimization problems, including cost estimation, cardinality estimation, and query optimization. However, the task of feature selection and encoding for machine learning in database tasks presents significant challenges. Recently, some representation methods have been proposed that utilize physical plan or SQL query as feature. However, these methods have two limitations. Firstly, they often rely on the selection of workloads using either the physical plan or the SQL query alone, which is not comprehensive enough to fully represent the characteristics of the database. Secondly, early feature extraction and encoding methods are not applicable to the database workload.

To tackle these limitations, we propose PQSEncoder, a feature representation model designed to address various database optimization challenges. In this approach, we integrate the physical plan, SQL query, and database schema to construct the workload of the database. Feature extraction and encoding are performed for each type, followed by feature fusion to compose the workload's features, which can then be used for machine learning tasks in database optimization. We incorporate PQSEncoder into two machine learning models for database optimization tasks, and experimental results show that PQSEncoder substantially improves the performance of these models.

Keywords: physical plan · SQL query · database schema · AI4DB

1 Introduction

With the increasing complexity and diversity of modern database management systems (DBMS), the optimization and maintenance of databases have become more intricate. Consequently, there has been a growing trend of applying machine learning techniques to various database tasks, including cardinality

estimation [5], join order selection [22], cost estimation [17], and performance optimization [10, 24]. Feature engineering plays a critical role in the training process of machine learning models, where the workload serves as the input feature for machine learning in databases. The selection, feature extraction, and encoding of the workload lay the foundation for effectively applying machine learning techniques to address database tasks.

In existing database machine learning tasks, the physical plan is predominantly utilized as workload features [5, 10, 22, 24]. Describing a sequence of operations and their corresponding costs during query execution, the physical plan is commonly employed for machine learning models. Additionally, other features such as the SQL query and database schema are utilized as workload features [19]. The SQL query contains rich semantic and structural information about the query, while the database schema provides metadata and data distribution information.

Although current research has made strides in addressing feature engineering challenges in database machine learning, it still faces limitations. Firstly, existing methods often select only a subset of features to represent the workload, which may not comprehensively and accurately characterize the database workload. For example, aggregate operations in a physical plan may lack specificity regarding whether they involve summation, averaging, or finding the maximum value. Moreover, accurately extracting features for each type of workload remains a challenge. For instance, using a Convolutional Neural Network (CNN) model [11, 12] to encode the physical plan may struggle to capture long path information within the tree structure effectively. Many existing models rely on one-hot encoding to represent SQL queries [5, 17]. For instance, models like MSCN [17] use one-hot encoding to represent database tables and columns in queries for cardinality estimation. However, this approach has limitations as it may fail to preserve the semantic information of SQL queries.

To solve the identified challenges, this paper introduces QPSEncoder (QUERY-PLAN-SCHEMA-ENCODER), a versatile workload encoder. QPSEncoder integrates physical plans, SQL queries, and database schemas to construct comprehensive workload. The model conducts feature extraction and encoding for each feature type individually before amalgamating the encoded features. This unified feature representation is suitable for a variety of learning tasks. Moreover, QPSEncoder is integrated into multiple machine learning models to assess its efficacy in characterizing database workloads. Our key contributions include:

- 1) A dynamic feature extraction algorithm is designed to extract features from physical plans. Then utilizes tree convolutional neural networks and attention mechanisms to encode the physical plan.
- 2) We develop SQLBERT, a pre-trained SQL representation model, that excels at finding both the semantic and structural components within SQL queries, by utilizing self-attention techniques.
- 3) A graph structure is created by considering the column relationships with in the database schema and SQL query, with columns serving as vertices.

The database schema is then encoded using a variational graph autoencoder model.

- 4) Experimental validation confirms that the QPSEncoder can seamlessly integrate into database tasks. It has the potential to greatly improve the effectiveness of these machine learning methods in relation to database algorithms.

2 Related Work

To address the issue of workload characterization in database optimization tasks, different features of the database are selected as the features of the workload, and feature extraction and encoding are performed on these features to serve as inputs for machine learning. Physical plan is often used as inputs for some machine learning models, such as cardinality and cost estimation [17], index recommendation [2], query optimization [11, 12], view selection [23], and join order selection [22], etc. The Tree-RNN approach tackles the issue of adapting tree-like physical plans by following a hierarchical tree structure. RTOS [22] and TLSTM [17] utilize the Tree-LSTM model [18] to aggregate node information from leaf nodes to the root node in a bottom-up manner, generating the final output as the representation of the physical plan. However, due to their recursive nature, they are difficult to train with large physical plan. Tree-CNN [13] is an extension of traditional CNN [8] that allows for tree-like inputs. NEO [12] and BAO [11] use Tree-CNN with triangle filters to slide over the tree of physical plan, capturing the parent-child dependencies of the physical plan. However, these methods have smaller receptive fields and cannot capture long paths of information flow from leaf nodes to the root node. Yue Zhao proposed QueryFormer [25], a tree-based transformer model for physical plan representation, that uses physical plan as features to capture parent-child dependencies and long paths of information flow in physical plan.

Other features of databases can contribute to workload characterization as well. PreQR [25] introduces a novel pre-training SQL representation model that utilizes SQL queries as features. The model develops a new SQL encoder employing attention mechanisms to encode query structures using automata. Additionally, it encodes the schema definition of the database using a graph structure model. Query-aware subgraphs extract SQL-related pattern information, while attention mechanisms discern the relationship between the database graph schema and SQL structures. This representation treats the schema structure as a graph, with each table represented as a node and foreign key relationships as edges. Another approach, SpiderSchemaGnn [1], constructs the database schema as a graph structure. It then employs graph neural networks to learn representations of nodes and edges, thereby capturing comprehensive and rich database structural information.

3 QPSEncoder Framework

We propose a general encoding framework QPSEncoder, as shown in Fig. 1. To characterize the characteristics of a database during queries execution, we utilize

the physical plan P , SQL query Q , and database schema S to form a workload w represented as:

$$w = \{Q, P, S\} \tag{1}$$

Firstly, we propose a dynamic feature extraction algorithm to extract features from physical plans. The extracted features are encoded using single-node encoding and tree-structure encoding. The encoded tree-structure feature vectors are further encoded into fixed-size vector representations P_v using tree-CNN and spatial attention models, as shown in Sect. 3.1.

Secondly, for SQL queries encoding, we design a tokenizer that is suitable for both databases and SQL queries. We propose an SQL queries encoding model called SQLBert. Additionally, we incorporate structural information of SQL queries into SQLBert. Finally, the encoded SQL queries is further encoded into a fixed-size vector representation Q_v using neural networks, as described in Sect. 3.2.

Finally, for database schema encoding, we employ a combination of static and dynamic graphs to construct a graph structure that links the database schema with the SQL query structures. We use the Variational graph auto-encoders(VGAE) [7] model to extract and encode features from the constructed graph structure, resulting in a fixed-length vector representation of the graph structure, denoted as S_v , as described in Sect. 3.3. We then merge the three feature vectors together to form the workload features Y_v :

$$Y_v = Concat(P_v, Q_v, S_v) \tag{2}$$

which serves as input for database machine learning tasks.

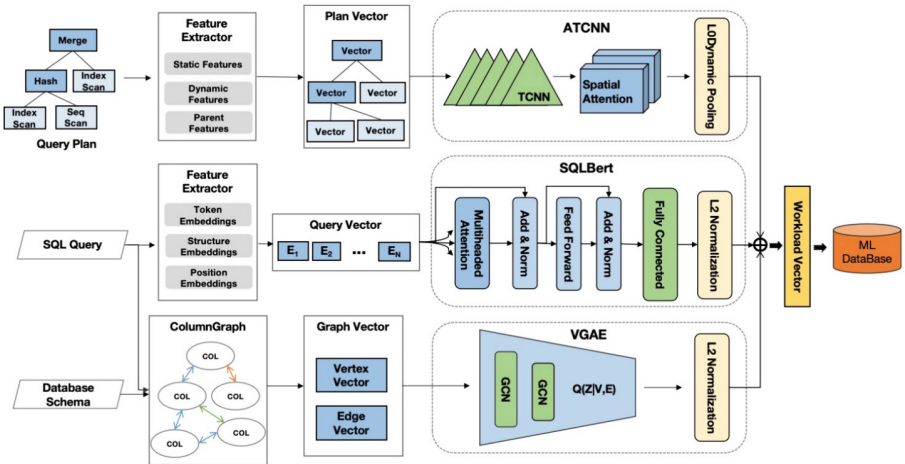


Fig. 1. The architecture of QPSEncoder.

3.1 Physical Plan Encoding

The node of query physical plan typically contain information such as operators, relations, predicates, cardinality and cost estimates. We select operators and attributes of numeric types as features for the physical plan. Other non-numeric types are encoded using SQL queries and database schema to extract features. For example, relations can be encoded using SQL queries and database schema, and predicates can be encoded using SQL queries. Considering both node information and parent-child dependencies, we categorize the physical plan features into three types: static features, dynamic features, and parent features. Static features include frequently used operators in the physical plan. Dynamic features include the dynamic characteristics and resource usage of SQL queries during execution, such as Total Cost, Plan Rows, and Shared Hit Blocks. We select features with high level of discreteness as dynamic features, as they have better performance. Parent features include all operators in the physical plan.

Operators are chosen as features by static features according to their usage rate, which is defined as follows:

$$Fuseage(operator) = (usedCount(operator) \div \sum_{d \in D} \sum_{n \in N} Count(*)) \times 100\% \quad (3)$$

$usedCount(operator)$ is the number of a operator used, D is a queries dataset, and N is a physical plan tree. Static features chose appropriate operators as static features by adjusting the usage threshold, which is obtained by aggregating all operators from physical plan nodes in the queries dataset.

Dynamic features evaluate the discreteness of the feature through variance, which is calculated by computing the variance of each dynamic attribute in the physical plan nodes, and is defined as follows:

$$Fvariance(element) = \Sigma(x_i - \mu(element))^2 \div N \quad (4)$$

In a queries dataset, $\mu(element)$ is the average value of a element all physical plan nodes in the query dataset D , and N is the number of physical plan nodes in the query dataset D . Appropriate dynamic features are chosen by adjusting the threshold of variance. Finally, to extract the parent-child relationships of each node in the physical plan, we form the parent features by grouping all operators that appear in the physical plan, and encode the parent operator into the child nodes.

We encode three types of features: parent features and static features are encoded using one-hot encoding [15], while dynamic features are encoded as normalized numerical values. The specific encoding algorithm is described in Algorithm 1. For parent feature encoding, the operator of the parent node is encoded into the parent features of the child node using one-hot encoding. The operator type of the parent node is encoded as 1 in the child node's parent feature, while other operators are encoded as 0. Static features are encoded using one-hot encoding, where if a certain node type belongs to the static features, it is encoded as 1, and the rest of the operators are encoded as 0. Dynamic features themselves are numerical types, and their numerical values are directly

used as feature values. However, the significant difference between the numerical values of dynamic features and one-hot encoding values may affect the training of classifiers [3]. Therefore, we perform min-max normalization on the numerical values of dynamic features to unify the values within the range of [0,1]. The three types of features constitute a feature vector for each node, and the complete feature vector tree is formed among the nodes through parent features.

Algorithm 1. Encoding of a physical plan.

Input: a set of static feature S , a set of dynamic feature D , a set of parent feature P , a plan tree N

Output: a vertex of a physical plan tree Fv

```

1: Initialize static features vector  $Sv \leftarrow [0]$ , dynamic feature vector  $Dv \leftarrow [0]$ , parent
   feature vector  $Pv \leftarrow [0]$ 
2: for  $planNode \in N$  do
3:   if  $planNode$  is not root then
4:      $parentNode \leftarrow planNode(parentNode)$ 
5:      $Pv[parentNode(nodeType)] \leftarrow 1$ 
6:   end if
7:   if  $planNode(nodeType) \in S$  then
8:      $Sv[planNode(NodeType)] \leftarrow 1$ 
9:   end if
10:  for  $element \in planNode$  do
11:    if  $element(name) \in D$  then
12:       $Dv[element(name)] \leftarrow element(value)$ 
13:    end if
14:  end for
15:   $Dv \leftarrow \|Dv\|$ 
16:   $Fnode \leftarrow Sv \oplus Dv \oplus Pv$ 
17:   $Fv$  add  $Fnode$ 
18: end for
19: return  $Fv$ 

```

Obtaining a representation of the physical plan tree vector poses challenges because existing methods have difficulty capturing all important information. Firstly, the Tree-CNN method [11, 12] uses average or dynamic pooling to aggregate features from nodes. These pooling operations down sample information in a brute-force manner, leading to information loss. We enhance the attention given to important features by adding an attention mechanism to the classification model. Spatial attention mechanism [20] is an attention module for CNN that enhances the network’s focus on important features and suppresses the influence of irrelevant features. In this way, the spatial attention mechanism can improve the network’s ability to extract essential features and enhancing its performance. Therefore, we integrate the spatial attention mechanism into the tree-CNN named ATCNN.

The ATCNN sub-model architecture consists of convolutional and fully connected layers. The convolutional layer comprises five tree convolutions and two

spatial attention mechanisms. The feature tree vector first enters a layer of tree convolution with a dimension of length 256. The convolution kernel of each tree convolution is 3, and each layer of tree convolution undergoes layer normalization before undergoing spatial feature extraction through a layer of spatial attention mechanism. The output length of the four subsequent tree convolutions are 128, 64, 32, and 16, respectively. The attention mechanism is applied to further extract features after the last convolutional layer, and the dynamic pooling is used to convert the tree structure into a vector of length 16.

3.2 SQL Query Encoding

Bidirectional Encoder Representations from Transformers (BERT) [4] stands as one of the most widely used pre-trained models in natural language processing (NLP). It’s a Transformer-based sentence encoder that calculates attention weights using multiple independent attention heads, enhancing representations. BERT is unsupervised trained on a huge corpus of text and learns rich sentence representations that make it applicable to various NLP tasks.

We pruned and modified the BERT model. In the original BERT, tokenization is used to handle multiple sentences as input, where each sentence is encoded as a distinct segment. However, the SQL query consist of a single statement and do not require segmentation. Instead, we replaced segment encoding with structural encoding specific to SQL query. Positional, structural, and token encoding are used to create a composite embedding for every token in a SQL query. We give an example of a SQL embedding in Fig. 2. In classification jobs, the classification token [CLS] serves as an aggregate representation of the entire token sequence and is the first token in the SQL query. The SQL query comes to a close with the [END] token. The composite embeddings from the three encodings are fed into the SQLBERT module to provide a fixed-length vector representation of the SQL query.

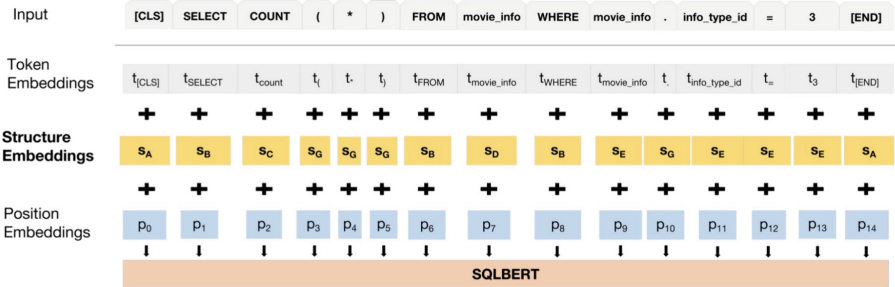


Fig. 2. SQLBERT model Input.

In a SQL query, the same table or column name can appear in the SELECT, WHERE, ORDER BY, and GROUP BY structure simultaneously. While

BERT’s positional encoding captures relative positional information between words in input sentences, the positions of table and column names in SQL queries vary, making positional encoding unsuitable for accurately representing the same table and column names across different positions. To address this, we categorized SQL queries into 15 structural types (see Table 1), differentiating between keywords, symbols, and other words, as well as table and column names in various structures. Each structure type is encoded as depicted in Fig. 3. For example, s_B represents keywords, and s_E represents table name, column name and operator in the WHERE structure.

Table 1. Structure types of SQL queries.

| SQL’s Structure | Example | Code |
|-----------------------|----------------------------------|------------------------|
| SQL START AND END | [CLS], [END] | 0000 |
| KEY WORD | select, update, insert, from ... | 0100 |
| SELECT STRUCTURE | table, column, column alias | 0201, 0202, 0203 |
| FROM STRUCTURE | table, table alias | 0301, 0302 |
| WHERE STRUCTURE | table, column, operator, value | 0401, 0402, 0403, 0404 |
| HAVING STRUCTURE | table, column, operator, value | 0501, 0502, 0503, 0504 |
| ORDER BY STRUCTURE | table, column | 0601, 0602 |
| GROUP BY STRUCTURE | table, column | 0701, 0702 |
| LIMIT STRUCTURE | table, column | 0801, 0802 |
| INSERT INTO STRUCTURE | table, column, value | 0901, 0902, 0903 |
| UPDATE STRUCTURE | table, column, value | 1001, 1002, 1003 |
| DELETE STRUCTURE | table, column, value | 1101, 1102, 1103 |
| ON STRUCTURE | table, column, operator, value | 1201, 1202, 1203, 1204 |
| FUNCTION STRUCTURE | count, sum.max, min... | 1300 |
| SYMBOL STRUCTURE | “.”, “,”, “(”, “)”, “*” | 1400 |

Pre-training of SQLBERT model: SQL queries contain a vast number of database-specific terminology, is different natural language. WordPiece [21] embeddings are used by the BERT model to split down original words into smaller subwords and characters. For example, the field “info_type_id” would be divided into five subwords: “info”, “_”, “type”, “_” and “id”. To maintain specialized terminology such as table names and column names, we created a bespoke tokenization technique based on the structure and vocabulary of SQL queries. In addition, as training data, we use 80,000 words of database keywords and database metadata. Using the masked word prediction task, we train the model by retraining it on top of a previously trained model. 15% of the input tokens are randomly masked in this challenge, and the model is trained to predict these masked tokens. Each word is encoded into a one dimensional vector of length 768 after the encoding transformer layer. The feature vector length is then reduced to 16 by two fully connected layers, and an L2 normalization layer is applied to generate a normalized vector of length 16.

3.3 Database Schema Encoding

Existing database schema is encoded as graph structure, mostly constructed by using tables as vertices and primary-foreign key relationships as edges [25]. This method lacks the ability to capture dynamic information during queries and cannot capture column correlations. We model columns and their correlations as a graph, where vertices represent columns. Each vertex contains the primary data and queries features corresponding to the column. We encode the features of each vertex into a vector with 7 dimensions: table id, table size, tuple selectivity, tuple length, whether index, average width and n-distinct.

In the relationship between columns, we consider both the static relationships in the database schema and the dynamic relationships that connect columns in the queries. The static relationship includes two columns belonging to the same table R_1 and a primaryforeign key relationship R_2 , the dynamic relationship is the columns in a query are connected through a join operation R_3 . The column relationships are denoted as $R = \{R_1, R_2, R_3\}$. The graph structure is defined as $G_s = \{V, E, R\}$, where $v_i \in V$ represents a node, $e_i \in E$ represents an edge, and $r \in R$ represents the relationship type. E represents 3 types of edges. For each pair of nodes v_x and v_y in the graph, Fig. 3 describes how to create an edge (v_x, v_y, r) , where r is the label of the edge.

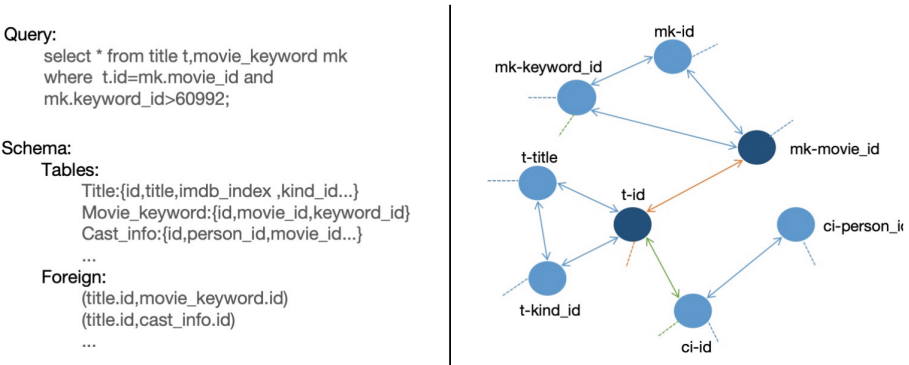


Fig. 3. The column graph.

Most existing graph encoding algorithms directly employ graph convolutional neural networks (GCNs) [6] for feature extraction of graph structures. However, using GCNs for graph encoding requires the construction of labels for supervised learning, making the training process challenging. Auto-encoders (AE) [14], on the other hand, are unsupervised learning models used to learn efficient representations from input data, thus reducing training difficulties and improving model generalization. They consist of an encoder and a decoder.

The VGAE model applies Autoencoder to the graph domain. Instead of being obtained from a deterministic GCN, node vectors are sampled from a

multi-dimensional Gaussian distribution. Node embedding vectors are sampled from a multi-dimensional Gaussian distribution as well. By uniquely determining a multi-dimensional Gaussian distribution through μ and δ , samples can be drawn from it to obtain the embedding representation of nodes. The posterior probability distribution of the embedding vectors is:

$$q(Z|V, E) = \prod_{i=1}^N q(z_i|V, E) \quad (5)$$

include, $q(z_i|V, E) = N(z_i|\mu_i, \text{diag}(\delta^2))$, μ and δ determined by two-layer GCNs.

The decoder reconstructs the graph by calculating the probability of edges between two points:

$$q(E|Z) = \prod_{i=1}^N \prod_{j=1}^N p(E_{ij}|z_i, z_j) \quad (6)$$

The VGAE model uses reconstruction loss and Kullback-Leibler (KL) divergence loss to train the model. Reconstruction loss measures the difference between the graph structure generated by the decoder and the original graph structure, while KL divergence loss measures the difference between the distribution of latent variables and the standard normal distribution:

$$L = E - q(Z|V, E)[\log p(V|Z)] - KL[q(Z|V, E)||P(Z)] \quad (7)$$

Pre-training of VGAE model: We utilize the IMDB database schema and 10,000 SQL statements, with each SQL query and the database schema forming a graph structure. The vertex matrix V and edge matrix E are obtained as inputs for the VGAE model. Adam optimizer is used in VGAE, and the training continues either until reaching 200 batches or achieving convergence (measured by a decrease in training loss of less than 1% within 10 batches). Finally, encode the two input matrices into a vector of length 16.

4 Experiments

To evaluate the effectiveness of the QPSEncoder model, we conducted experiments on two machine learning tasks: cost estimation and cardinality estimation in database. All of our work is performed on a Linux server with specific configuration parameters as follows: CentOS Linux release 7.9.2009 (Core) 64-bit Operating System, 40-core Intel(R) Xeon(R) Silver 4210R CPU @ 2.40 GHz, and NVIDIA GeForce RTX 3080 GPU. We use PostgreSQL version 14.0 as the database for generating physical plans. For implementing neural network models, we employ PyTorch, and models are executed using Python 3.7 interpreter.

4.1 Experimental Setup

Training Datasets. For queries cardinality and cost estimation, we employed the IMDB dataset, where columns and tables exhibit high correlation, making it highly challenging for all types of database tasks. The dataset consists of 22 tables interconnected through primary and foreign key relationships. We used a subset of 10,000 queries from the real IMDB dataset as training data, The 10,000 queries are split to training and validation set with 9 : 1 ratio.

Verify Datasets. The first workload consists of predicates with numerical attributes. We choose two of sub-workloads that only contain numerical predicates: synthetic workload Scale with 500 queries, aimed at demonstrating how the model generalizes to more joins; JOB-light, a workload derived from the Join Order Benchmark (JOB) [9], comprising 70 queries that do not involve any predicates on strings or disjunctions, and are limited to a maximum of four joins.

The second workload is derived from the JOB benchmark, where queries involve complex predicates on string attributes, totaling 113 queries. The range of join counts for the queries in the JOB workload is between 4 and 28.

Comparison Baselines. For the two estimation tasks, we used the following four methods as baselines: (1) PostgreSQL (PG): PostgreSQL uses statistical and cost models for cardinality and cost estimation. (2) MSCN: A CNN-based model is used for queries-level cardinality and cost estimation [17]. (3) TLSTM: A model based on LSTM is trained for queries cost estimation and can also be used for queries cardinality estimation [5]. (4) QueryFormer: A tree-structured transformer-based physical plan representation model that can be used for cardinality estimation and cost estimation [19].

To showcase the effectiveness of the encoding by the QPSEncoder, we employ a very simple fully connected model as our prediction model. For each query, its workload’s (1 * 48) encoding generated by QPSEncoder is input into a 2-layer fully connected feature extractor, and the final prediction result is generated using sigmoid. In physical plan encoding, the usage threshold of static features is set to 0.6, and the variance threshold of dynamic features is set to 1000000.

Evaluation Metrics. According to [16], we use the Pearson correlation coefficient between predicted value and actual value on a logarithmic scale. This measures the goodness of fit between the predicted values and the actual values.

$$qerror(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^n \frac{\max(y_i, \hat{y}_i)}{\min(y_i, \hat{y}_i)} \quad (8)$$

4.2 Effectiveness of QPSEncoder on Numeric Predicates.

We train the models and test them on workloads with numeric predicates only, specifically Scale and JOB-light. The cost estimation results for JOB-light and Scale are presented in Table 2, while the cardinality estimation results for the same workloads are shown in Table 4. In both cost and cardinality estimation

Table 2. Cost errors on numeric workloads.

| JOB-light | median | 90th | 95th | 99th | max | mean |
|-------------|-------------|-------------|--------------|---------------|---------------|-------------|
| PG | 26.8 | 332 | 696 | 2740 | 3020 | 173 |
| MSCN | 4.75 | 11.3 | 40.1 | 563 | 987 | 27.4 |
| TLSTM | 3.66 | 32.1 | 80.3 | 445 | 583 | 17 |
| QueryFormer | 1.4 | 20.27 | 50.19 | 176.59 | 379.97 | 12.49 |
| PSEncoder | 16.39 | 237.80 | 440.39 | 1254.95 | 1444.90 | 106.39 |
| QPSEncoder | 7.30 | 110.11 | 307.07 | 799.60 | 1673.59 | 61.15 |
| QSEncoder | 10.70 | 121.31 | 158.10 | 452.80 | 690.84 | 41.11 |
| QPSEncoder | 2.61 | 18.6 | 28.05 | 106.15 | 108.57 | 8.35 |
| scale | median | 90th | 95th | 99th | max | mean |
| PG | 13.3 | 38.9 | 81.1 | 718 | 1473 | 35.7 |
| MSCN | 1.79 | 10.6 | 27.1 | 88.8 | 1027 | 8.22 |
| TLSTM | 1.58 | 5.51 | 14.4 | 70.1 | 611 | 5.21 |
| QueryFormer | 1.43 | 5.18 | 16.63 | 71.58 | 588.20 | 5.03 |
| PSEncoder | 5.74 | 163.97 | 495.08 | 1994.62 | 6474.51 | 107.95 |
| QPSEncoder | 8.67 | 65.75 | 155.74 | 800.43 | 1562.33 | 43.98 |
| QSEncoder | 5.95 | 56.48 | 122.00 | 401.25 | 832.68 | 29.35 |
| QPSEncoder | 1.24 | 4.25 | 11.74 | 64.12 | 158.9 | 4.78 |

Table 3. Cost errors on the JOB workload.

| Method | median | 90th | 95th | 99th | max | mean |
|------------|-------------|-------------|-------------|--------------|--------------|-------------|
| PG | 4.90 | 80.8 | 104 | 3577 | 4920 | 105 |
| TLSTM | 4.01 | 14.9 | 24.5 | 105 | 148 | 9.4 |
| QPSEncoder | 3.84 | 20.43 | 28.9 | 72.78 | 89.35 | 9.24 |

tasks, QPSEncoder demonstrates significant improvement. On the JOB-light workload, QPSEncoder demonstrates significant improvements in both cardinality and cost estimation accuracy compared to the other four methods. It achieves an average error reduction of 30% in cardinality estimation compared to QueryFormer and nearly 50% in cost estimation. This indicates that QPSEncoder, by capturing query structure, statements, and database schema information, outperforms QueryFormer, which relies solely on physical plan and statistical information. Additionally, QPSEncoder also shows varying degrees of improvement on the Scale workload, indicating that QueryFormer has the ability to generalize and adapt to changing workloads.

Table 4. Cardinality errors on numeric workloads.

| JOB-light | median | 90th | 95th | 99th | max | mean |
|-------------|-------------|--------------|--------------|---------------|---------------|--------------|
| PG | 7.93 | 164 | 1104 | 2912 | 3477 | 174 |
| MSCN | 3.82 | 78.4 | 362 | 927 | 1110 | 57.9 |
| TLSTM | 3.73 | 50.8 | 157 | 256 | 289 | 24.9 |
| QueryFormer | 2.26 | 38.74 | 225 | 326 | 373 | 29.50 |
| PSEncoder | 6.20 | 32.18 | 87.02 | 157.94 | 214.66 | 17.51 |
| QPEncoder | 2.83 | 29.37 | 108.04 | 447.84 | 493.63 | 25.90 |
| QSEncoder | 26.31 | 493.19 | 859.36 | 2334.34 | 3983.88 | 199.60 |
| QPSEncoder | 7.64 | 30.4 | 49.07 | 126.5 | 170.31 | 15.42 |
| scale | median | 90th | 95th | 99th | max | mean |
| PG | 2.59 | 200 | 540 | 1816 | 233863 | 568 |
| MSCN | 1.42 | 37.4 | 140 | 793 | 3666 | 35.1 |
| TLSTM | 1.43 | 38.8 | 139 | 469 | 1892 | 28.1 |
| QueryFormer | 1.40 | 39.2 | 128 | 414 | 1748 | 26.9 |
| PSEncoder | 4.73 | 58.00 | 174.87 | 1068.36 | 9303.21 | 62.79 |
| QPEncoder | 2.22 | 18.77 | 38.14 | 271.16 | 13809.80 | 55.81 |
| QSEncoder | 13.72 | 285.75 | 855.78 | 18832.88 | 81471.20 | 658.19 |
| QPSEncoder | 1.38 | 38.52 | 70.53 | 485.03 | 1901.76 | 26.26 |

Table 5. Cardinality errors on the JOB workload.

| Method | median | 90th | 95th | 99th | max | mean |
|------------|-------------|-------------|--------------|--------------|---------------|-------------|
| PG | 184 | 8303 | 34204 | 1.06e5 | 6.70e5 | 10416 |
| TLSTM | 10.1 | 130 | 223 | 680 | 901 | 53.0 |
| QPSEncoder | 3.01 | 8.58 | 31.10 | 82.74 | 175.29 | 8.18 |

4.3 Effectiveness of QPSEncoder on Mixed Predicates.

We compared the PG, TLSTM, and QPSEncoder models on the JOB workload, testing them with both string and numeric predicates. The MSCN and QueryFormer models were excluded from this comparison due to their lack of support for string predicates. The cost estimation results are presented in Table 3, while the cardinality estimation results are shown in Table 5. Among the models, QPSEncoder outperformed both PG and TLSTM. Unlike TLSTM, which combines SQL keywords and regular predicates, QPSEncoder leverages SQLBERT to encode query statements. It tokenizes and encodes the predicates separately, thereby preserving both semantic and structural information of the queries.

4.4 Effectiveness of Model Components

To observe the impact of three features, on the model, we compared QPSEncoder with QPEncoder (no database schema), QSEncoder (no physical plan), and PSEncoder (no SQL query). Detailed results can be found in Table 2 and 4. In cost estimation, PSEncoder exhibits an average error reduction of almost 50% compared to QPEncoder and QSEncoder, while QPEncoder and QSEncoder display similar average errors. This highlights the substantial impact of the SQL query on the model’s performance in cost estimation tasks. In cardinality estimation, QSEncoder demonstrates an average error reduction of nearly 10 times compared to QPEncoder and QSEncoder, while QPEncoder and QSEncoder showcase similar average errors. This underscores the significant influence of the physical plan on the model’s performance in cardinality estimation tasks. Across all metrics, QPSEncoder outperforms others, emphasizing the importance of integrating all three features to more accurately represent the characteristics of the database workload.

5 Conclusion

In this paper, we address the issue of database workload representation, a fundamental component for machine learning algorithms in databases. We introduce QPSEncoder, a versatile encoder designed to handle three key features: the physical plan, SQL query, and database schema. Utilizing advanced algorithms, QPSEncoder extracts and encodes features from each input based on their specific characteristics. Through extensive experiments conducted on two machine learning tasks for databases and two real datasets, we evaluate the effectiveness of QPSEncoder. The results demonstrate that QPSEncoder enhances the performance of existing methods by providing a more efficient workload representation.

Acknowledgements. This work is supported by the Key R&D Program of Shandong Province under Grant 2021CXGC010104, and the National Key R&D Program of China under Grant 2022YFF0503900. Author Jin Yan would like to thank the ANSO Scholarship for Young Talents for sponsorship during prusuphdng her PhD degree.

References

1. Bogin, B., Berant, J., Gardner, M.: Representing schema structure with graph neural networks for text-to-SQL parsing. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 4560–4565 (2019)
2. Ding, B., Das, S., Marcus, R., Wu, W., Chaudhuri, S., Narasayya, V.R.: AI meets AI: leveraging query executions to improve index recommendations. In: Proceedings of the 2019 International Conference on Management of Data, pp. 1241–1258 (2019)
3. Juszczak, P., Tax, D., Duin, R.P.: Feature scaling in support vector data description. In: Proceedings of the ASCI, pp. 95–102. Citeseer (2002)

4. Kenton, J.D.M.W.C., Toutanova, L.K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of naacL-HLT, vol. 1, p. 2 (2019)
5. Kipf, A., Kipf, T., Radke, B., Leis, V., Boncz, P., Kemper, A.: Learned cardinalities: Estimating correlated joins with deep learning. In: 9th Biennial Conference on Innovative Data Systems Research (CIDR 2019) (2019)
6. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint [arXiv:1609.02907](https://arxiv.org/abs/1609.02907) (2016)
7. Kipf, T.N., Welling, M.: Variational graph auto-encoders. arXiv preprint [arXiv:1611.07308](https://arxiv.org/abs/1611.07308) (2016)
8. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: NIPS, vol. 25 (2012)
9. Leis, V., Gubichev, A., Mirchev, A., Boncz, P., Kemper, A., Neumann, T.: How good are query optimizers, really? Proc. VLDB Endow. **9**(3), 204–215 (2015)
10. Li, G., Zhou, X., Li, S., Gao, B.: QTune: a query-aware database tuning system with deep reinforcement learning. Proc. VLDB Endow. **12**(12), 2118–2130 (2019)
11. Marcus, R., Negi, P., Mao, H., Tatbul, N., Alizadeh, M., Kraska, T.: Bao: making learned query optimization practical. In: Proceedings of the 2021 International Conference on Management of Data, pp. 1275–1288 (2021)
12. Marcus, R., et al.: Neo: a learned query optimizer. Proc. VLDB Endow. **12**(11), 1705–1718 (2019)
13. Mou, L., Li, G., Zhang, L., Wang, T., Jin, Z.: Convolutional neural networks over tree structures for programming language processing. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30 (2016)
14. Rifai, S., Vincent, P., Muller, X., Glorot, X., Bengio, Y.: Contractive auto-encoders: explicit invariance during feature extraction. In: Proceedings of the 28th International Conference on International Conference on Machine Learning, pp. 833–840 (2011)
15. Seger, C.: An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing (2018)
16. Siddiqui, T., Jindal, A., Qiao, S., Patel, H., Le, W.: Cost models for big data query processing: learning, retrofitting, and our findings. In: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, pp. 99–113 (2020)
17. Sun, J., Li, G.: An end-to-end learning-based cost estimator. Proc. VLDB Endow. **13**(3)
18. Tai, K.S., Socher, R., Manning, C.D.: Improved semantic representations from tree-structured long short-term memory networks. 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), pp. 1556–1566 (2015)
19. Tang, X., Wu, S., Song, M., Ying, S., Li, F., Chen, G.: PreQR: pre-training representation for SQL understanding. In: Proceedings of the 2022 International Conference on Management of Data, pp. 204–216 (2022)
20. Woo, S., Park, J., Lee, J.Y., Kweon, I.S.: CBAM: convolutional block attention module. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 3–19 (2018)
21. Wu, Y., et al.: Google’s neural machine translation system: bridging the gap between human and machine translation. arXiv preprint [arXiv:1609.08144](https://arxiv.org/abs/1609.08144) (2016)

22. Yu, X., Li, G., Chai, C., Tang, N.: Reinforcement learning with tree-LSTM for join order selection. In: 2020 IEEE 36th International Conference on Data Engineering (ICDE), pp. 1297–1308. IEEE (2020)
23. Yuan, H., Li, G., Feng, L., Sun, J., Han, Y.: Automatic view generation with deep learning and reinforcement learning. In: 2020 IEEE 36th International Conference on Data Engineering (ICDE), pp. 1501–1512. IEEE (2020)
24. Zhang, J., et al.: An end-to-end automatic cloud database tuning system using deep reinforcement learning. In: Proceedings of the 2019 International Conference on Management of Data, pp. 415–432 (2019)
25. Zhao, Y., Cong, G., Shi, J., Miao, C.: QueryFormer: a tree transformer model for query plan representation. *Proc. VLDB Endow.* **15**(8), 1658–1670 (2022)



Efficient Random Sampling from Very Large Databases

Idan Cohen^(✉), Aviv Yehezkel, and Zohar Yakhini

Reichman University, Herzliya, Israel

idan.cohen@post.runi.ac.il, aviv@yehezkel.net, zohar.yakhini@runi.ac.il

Abstract. One of the major research questions in large databases is how to efficiently sample a random subset of records. This sample can then be used to estimate query results and optimize query execution plans and other tasks. In order to have quick access to the data, the common practice is to create an index, which is often implemented by using B+Trees. Existing state-of-the-art algorithms for random sampling over B+Trees result in a significant performance overhead. This paper proposes novel approaches for efficient random sampling over B+Trees in very large databases. We analyze the algorithms' correctness and use extensive simulation study, which showcases their superior performance compared to previous works while not affecting the quality of the random sample.

Keywords: Large Databases · Query Processing · B+Trees · Random Sampling

1 Introduction

A common practice for solving tasks in real-world Big-Data applications is sampling a random subset of the data and using it to infer quantities of interest, assuming it is a good representation of the features of the full data. The benefits of working on a small sample of the data are priceless: speeding up computation time, reducing computation costs, saving resources, etc. [Minkkinen (2004), Liu and Zhang (2020)]. Some random sampling usage examples are data profiling [Abedjan et al. (2015)], data analysis [Slavakis et al. (2014)], data mining [Wu et al. (2013)], data visualization [Agrawal et al. (2015)] and machine learning applications [Papaemmanouil et al. (2016)].

In this work, we address the task of uniform random sampling over B+Trees [Comer (1979)], which is the primary mechanism for implementing indices in relational database systems that allow quick data retrieval [Kudale (n.d)]. Specifically, random sampling is commonly used in databases as an initial step for a variety of tasks such as optimizing queries, choosing execution plans, estimating aggregate query results (e.g., SUM, COUNT, or AVERAGE), and estimating query results [Piatetsky-Shapiro and Connell (1984), Naughton and Seshadri (1990), Hou et al. (1991), Lipton et al. (1993), Haas et al. (1994), Chaudhuri

et al. (1998)]. While sampling could be either uniform or non-uniform, in this paper, we study uniform sampling, where each population element has an equal probability of inclusion in the sample, providing a representative sample of the population as a whole.

The problem of efficient sampling from databases was extensively studied in the past [Olken and Rotem (1989), Haas (2003), Olken and Rotem (1995), Vitter (1985)]. In particular, the classical work of Olken [Olken and Rotem (1989)] from the late 80s has remained one of the prominent solutions to date. Olken’s approach is based on performing iterative random walks from the B+Tree’s root to its leaves and deciding whether to sample the given leaf or not according to an “Acceptance/Rejection (A/R) Test”. By adjusting the random walk, Olken’s approach ensures each leaf will be sampled with the same probability. Thus, the resulting sample represents a uniform distribution of the values stored in the tree leaves.

Though simple and elegant, as will be clearly shown in our simulation study, Olken’s approach is not satisfying in terms of efficiency. It may be impractical in real-world applications with very large databases [Haas (2003), Chaudhuri et al. (1998)]. The main drawback is the need to run many iterative random walks until a leaf is accepted to be sampled, which leads to a significant overhead.

Improvement attempts could not solve this efficiency issue, and the state-of-the-art algorithms for B+Tree sampling are still based on the A/R method. For example, the “Early Abort Method” [Olken and Rotem (1989)] suggested running the Acceptance/Rejection Test in each node along the path instead of waiting until reaching the leaf. The intuition is that a path from the root to the leaf can be rejected without retrieving the entire path, thus speeding up the iterations. Indeed, the expected cost of the early abort method is significantly less than Olken’s original method [Olken and Rotem (1989)]. However, the rejections have not vanished, and it still requires reading many more disk blocks compared to the sample size. Assuming we are sampling 1,000 values from a specific B+Tree of height four¹, using Olken’s Theorems 1 and 2 [Olken (1993)], the expected number of disk blocks that will be read is 64,000 for the original algorithm and 30,000 for the early abort method. While providing an improvement of 53% compared to the original Olken’s algorithm, the early abort method still has a significant 30x overhead for returning as few as 1,000 values.

To demonstrate the inefficiency of Olken’s algorithm, we conducted a series of experiments on a B+Tree of 1 million values. For example, we noticed that when sampling 10,000 rows from a B+Tree of 1 million values, Olken’s A/R test rejected 450,000 candidates. Operating 460,000 random walks from the root to the leaves in a B+Tree of height four means that the algorithm had to perform around two million data accesses.

The low performance of the existing state-of-the-art algorithms causes real-world applications running in commercial databases to prefer practical alternatives and heuristics instead, which also tend to suffer from low performance [Haas (2003)].

¹ Path’s length from the root to the leaves, with the root node being considered as height one.

The rest of this paper consists of the following: Sect. 2 describes related works. In Sect. 3, we present the new algorithms. In Sect. 4, we analyze the proposed algorithms and prove their correctness. In Sect. 5, we provide an extensive simulation study showing the superior performance of our proposed algorithms. Lastly, Sect. 6 concludes the paper.

2 Related Work

Random sampling from databases, and in particular B+Trees, is a common technique for solving data mining tasks such as Approximate Query Processing (AQP) [Li and Li (2018)], Chaudhuri et al. (2017)], online aggregation [Li et al. (2016)], Interactive Data Exploration [Papaemmanouil et al. (2016)], and more.

A straightforward approach involves performing random walks from the root of the B+Tree to its leaves and sampling a value randomly from the leaves. However, this intuition is wrong, as random walking does not provide an equal probability of picking each leaf. The lack of equal probability arises due to the differences in the number of children every non-leaf node has (practically stored as pointers to their children).

Recall that in B+Trees, the order m defines the maximum number of direct child nodes. Thus, the number of children of every non-leaf node is bounded by $\lceil \frac{m}{2} \rceil \leq |children| \leq m$. In order to point to its children, a node holds $(|children| - 1)$ pointers.

Thus, a value X under a leaf in a path with a node with $\frac{m}{2}$ children has a higher probability of being picked in a random walk compared to a value Y under a path with a node with m children, preventing a uniform sampling.

To address this, Olken [Olken and Rotem (1989)] suggested the ‘‘Acceptance/Rejection (A/R) Algorithm’’, which has remained a common practice to this date. Although Olken’s algorithm guarantees uniform sampling, its performance is far from satisfactory, and its running time overhead increases as the sample size grows. The low-performance results from Olken’s Acceptance/Rejection procedure: most of the candidates for sampling are actually being rejected, and thus, eventually, most of the data fetched from the disk is not used.

The state-of-the-art algorithms for B+Trees sampling are based on the A/R method. One of the latest works is the B+Tree Weighted Random Sampling (BTWRS) proposed by Makawita [Makawita et al. (2002)]. However, as described in Sect. 1 and will be shown in more detail in the simulation study in Sect. 5, the A/R-based algorithms suffer from long running times due to the inefficient rejection mechanism.

Several algorithms were proposed to outperform the A/R method, but they all require additional information: Olken [Olken and Rotem (1989)] adopted Wong’s algorithm [Wong and Easton (1980)] and suggested a method for random sampling from Ranked B+Tree. Although it achieves better results than the A/R method, the ranking might be expensive to maintain in a system with heavy updates. Antoshenkov [Antoshenkov (1992)] combined both methods, sampling over Ranked B+Tree and the A/R technique, and suggested the Pseudo-ranked

B+Tree (PRBT) sampling algorithm. However, despite the significant advantage made by Antoshkov in increasing the acceptance rate of the A/R mechanism to 50% [Antoshkov (1992)], the method still incurs a high cost and is restricted to specific variations of B+Trees that support essential compression and variable leaf page blocking [Olken and Rotem (1995)]. Nevertheless, a rejection rate of 50% still results in twice the number of operations required for a given sampling size. In contrast, our proposed method eliminates the need for any rejections.

For the general task of random sampling, regardless of the underlying data structure, two classical algorithms are “Reservoir Sampling” [Vitter (1985)] and “Bernoulli sampling” [Haas (2003)], which can be adopted and used for B+Tree random sampling as well. While these are generic methods that do not utilize the structure of a B+Tree, they do require fetching the entire database, which might be impractical.

3 The Proposed Algorithms

This section will present novel algorithms for random sampling from databases that will be more efficient than Olken’s and the A/R-based approaches. In Sect. 4, we will analyze the algorithms’ correctness and prove they provide an equal probability for every leaf to be sampled.

3.1 Random Sampling in B+Tree of Height Three

Algorithm 1 presents our proposed method for a B+Tree of height three. Instead of Olken’s random walk with an A/R test, the proposed approach is based on “intelligently-picked” random walks along the tree, such that each time a leaf is visited, its value will be sampled without any rejections. Thus, for sampling k values, the proposed algorithm will walk through exactly k paths². Namely, instead of performing an arbitrary random walk from the root to the leaf, our stochastic path selection will use the fanout distribution to correct the bias imposed by a uniform random walk (unlike Olken’s algorithms, which corrected this bias using the A/R test). Subsequently, in each iteration in which a leaf node is selected, the algorithm randomly chooses a record from that leaf. The random selection at the leaf level prevents any potential bias from being induced from the structure of the B+Tree, such as prioritizing the highest or lowest value in a leaf.

For example, Fig. 1 shows a B+Tree of height three. Following Algorithm 1 guarantees an equal probability ($1/6$) of reaching every leaf.

3.2 Random Sampling in B+Tree of Height Four

We now extend Algorithm 1 to support B+Trees of height four. To this end, we will define a new weights vector W_nodes , such that the step from the root to its

² In addition to possible paths that will be visited more than once, which is negligible in large B+Trees.

Algorithm 1. Weighted Sampling for B+Trees - Height three

```

sampled  $\leftarrow \square$ 
S  $\leftarrow \sum_{node \in children(root)} fanout(node)$ 
P  $\leftarrow [\frac{fanout(node_0)}{S}, \frac{fanout(node_1)}{S}, \dots, \frac{fanout(node_n)}{S}]$  {Pi denotes the probability to go
from the root to nodei}
while (len(sampled) < sample_size) do
    1. node  $\leftarrow$  randomly choose node from children(root) according to the probabili-
ties vector P.
    2. leaf  $\leftarrow$  randomly choose leaf from children(node) under uniform distribution.

    3. Randomly choose a value from leaf and add it to sampled.
end while

```

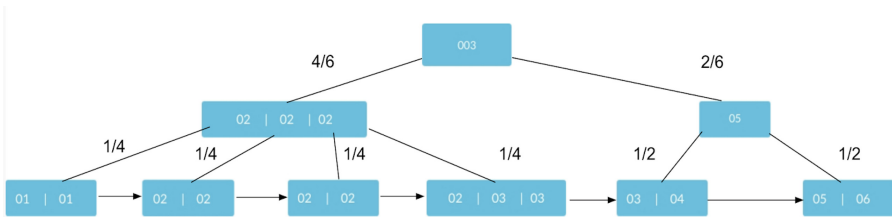


Fig. 1. Example for a tree with height three and how step probabilities are assigned according to Algorithm 1

children will be performed with respect to this weights vector. Afterward, since we will reach a sub-tree of height three, we will continue walking down the tree using Algorithm 1.

To achieve an equal probability of each leaf being selected in a B+Tree of height four, we construct the weights vector W_nodes such that each node’s weight is the proportion between its leaves to the entire B+Tree. To maintain the weights vector, we perform a one-time scan of the data, which requires $O(n)$ operations, and save a counter that holds the number of leaves found under each node. The weights vector is calculated directly from this counter. From now on, any B+Tree operation, which includes splitting or deleting a leaf, will require an additional $O(1)$ operation to update the counter. The complete method is presented in Algorithms 2 and 3.

For instance, consider the B+Tree of height four in Fig. 2. When following Algorithm 1 and counting the number of leaves at each sub-tree, we find seven leaves in the left sub-tree and four in the right. Therefore, the weights vector will be $W_nodes = [\frac{7}{11}, \frac{4}{11}]$. Then, using Algorithm 1 on the left sub-tree will associate all the leaves with the same probability of $\frac{1}{7}$ to be sampled (denoted by blue paths). Similarly, in the right sub-tree, each leaf will have the same probability of $\frac{1}{4}$ to be sampled (denoted by red paths). Therefore, using W_nodes vector as the first step, we obtain an equal probability of $\frac{1}{11}$ for every possible root-to-leaf path in the entire B+Tree:

Algorithm 2. Weighted Sampling for B+Trees - Height four

```

sampled ← []
while (len(sampled) < sample_size) do
  1. node ← randomly choose node from children(root) according to the weights
  vector W_nodes (see Algorithm 3).
  2. single_sampled ← random-sampling-height-3(btree = node, sample_size =
  1).
  3. Add single_sampled to sampled.
end while

```

Algorithm 3. Prepare Weights Vector *W_nodes*

```

for nodei ∈ children(root) do
  1. leaves_subtreei ← number of leaves in the sub-tree whose root is nodei.
end for
W_nodes ← [ $\frac{leaves\_subtree_0}{sum(leaves\_subtree)}$ , ...,  $\frac{leaves\_subtree_n}{sum(leaves\_subtree)}$ ]

```

- Path selection probabilities through a blue path: $\frac{7}{11} * \frac{1}{7} = \frac{1}{11}$
- Path selection probabilities through a red path: $\frac{4}{11} * \frac{1}{4} = \frac{1}{11}$

3.3 Generalization to Any B+Tree Height

Finally, we will show how Algorithms 1 and 2 can be generalized for B+Tree height higher than four. The main idea is that each node with more than two steps away from the leaves (the root in a B+Tree of height four is three steps away from the leaves) will be associated with a dedicated weights vector *W_nodes*. Thus, similarly to the weights vector of B+Tree of height four, for a given node α (which is more than two steps away from the leaves), $|W_nodes_\alpha| = |fanout(\alpha)|$, and $W_nodes_{\alpha_i} = \frac{leaves_subtree_i}{leaves_subtree_\alpha}$. Note that this equation is a generalization of the one suggested in Algorithm 3, and it holds that $\sum_{i \in W_nodes_\alpha} W_nodes_{\alpha_i} = 1$.

While this suggested generalization works in practice, its implementation may be complex as multiple *W_nodes* vectors need to be maintained, such that removal or addition of a leaf needs to propagate up to the root and update all the *W_nodes* vectors along the way. However, from a practical point of view, we are rarely required to maintain B+Trees of a height higher than four in real-world database applications. Recall that the B+Tree height is equal to $\log_m n$, where m is the order of the B+Tree, and n is the number of values. Thus, since the height is logarithmic in the number of values, and the fact that m in modern systems is usually in the hundreds [Graefe and Kuno (2011)], we can reach giant B+Trees with a height of four. For instance, a B+Tree of height four, with an order of 250 and a leaf size of 250, can store 4 billion elements.

Although the proposed Algorithms 1 and 2 yield equal probabilities at the leaf level and not at the record level, we will show in the simulation study in Sect. 5 that this gap is practically negligible.

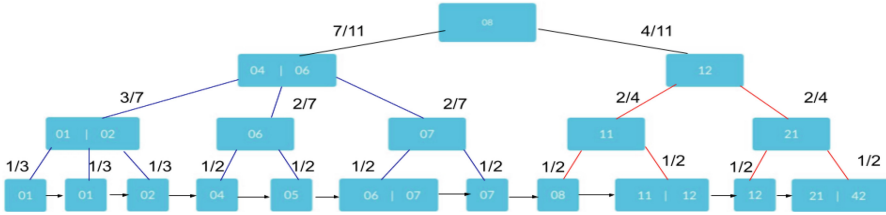


Fig. 2. Example for a B+Tree with height four and how step probabilities are assigned according to Algorithms 2 and 3.

4 Analysis

This section will analyze the proposed algorithms and prove their correctness.

Theorem 1. *For any B+Tree, the proposed algorithm provides an equal probability for every root-to-leaf path.*

For that, we first provide proofs for the algorithm of B+Tree Height three and Height four, and then we will prove the general case of any B+Tree with a height greater than four.

Proof. Height three:

Let *Path* be a root-to-leaf path generated by the algorithm.

Let f_{ij} be the fanout of the node denoted by the index j , in a level number i , such that, j is between 0 to the number of nodes in the tree in level i : $0 \leq j \leq nodes(level_i)$

Note that $level_0$ is the root, and thus $nodes(level_0) = 1$ in any B+Tree

Assume a specific *Path* chooses node number 0 on any tree level.

Then, according to the algorithm, the probability P of choosing this root-to-leaf *Path* will be:

$$P(Path) = \frac{f_{1,0}}{\sum_{j=0}^{nodes(level_1)} f_{1,j}} * \frac{1}{f_{1,0}} = \frac{1}{\sum_{j=0}^{nodes(level_1)} f_{1,j}}$$

Without loss of generality, this claim holds for any *Path* in the *B + Tree*. We can notice that $P(Path)$ is a fixed value, and thus the proposed algorithm provides an equal probability for every root-to-leaf path.

Proof. Height four:

Let $Path_0$ and $Path_1$ be two different root-to-leaf paths in a given B+Tree of height four.

Let f_{ij} be the fanout of the node denoted by the index j , in a level number i , such that, j is between 0 to the number of nodes in the tree in level i : $0 \leq j \leq nodes(level_i)$. Note that since $level_0$ is the root, then $nodes(level_0) = 1$ in any B+Tree. Then, since these paths also pass through $level_1$, we can write them as follows:

$\alpha =$ node in $level_1$

$\beta =$ node in $level_1$

$$\begin{aligned} Path_0 &= root \rightarrow \alpha \rightarrow node \in children(\alpha) \rightarrow leaf \\ Path_1 &= root \rightarrow \beta \rightarrow node \in children(\beta) \rightarrow leaf \end{aligned}$$

Recall that according to Theorem 1, all the paths from nodes α and to any leaf in this sub-B+Tree, whose root is α , have an equal probability of being selected.

Claim: $P(Path_0) = P(Path_1)$

If $\alpha = \beta$, then the first step of $Path_0$ and $Path_1$ has the same probability, and any walk to a leaf has the same probability, which follows from Theorem 1.

If $\alpha \neq \beta$, then let P_node_α be the probability of any root-to-leaf walk, using the suggested algorithm for B+Tree of height three, in a sub-tree whose root is α . Respectively, P_node_β is the probability of any root-to-leaf walk in a sub-tree whose root is β . According to the algorithm, any walk from $root$ to $children(root)$ is associated with a weight w , such that, each $node \in children(root)$ has a specific weight w_i .

Thus, the algorithm has calculated weights w_0, w_1 such that:

$$\begin{aligned} P(Path_0) &= w_0 * P_node_\alpha \\ P(Path_1) &= w_1 * P_node_\beta \end{aligned}$$

Notice that $w_0 * P_node_\alpha = w_1 * P_node_\beta$ according to W_nodes calculation in Algorithm 3.

We thus obtained that $P(Path_0) = P(Path_1)$ for any two paths in the B+Tree.

Proof. Generalization to any B+Tree height:

We will prove that by induction on the B+Tree height.

Induction Base: B+Tree Height five.

Let $Path_0$ and $Path_1$ be two root-to-leaf paths, generated by the algorithm, in a given B+Tree of height five, whose root is r . Assume α and β are the first nodes in the paths $Path_0$ and $Path_1$ respectively:

$$\begin{aligned} Path_0 &= r \rightarrow \alpha \rightarrow \dots \rightarrow leaf \\ Path_1 &= r \rightarrow \beta \rightarrow \dots \rightarrow leaf \end{aligned}$$

Recall the weights vector W_nodes_r from Sect. 3.3. Thus, in path $Path_0$, the first step was made by choosing node α with probability $\frac{leaves_subtree_\alpha}{leaves_subtree_r}$. Similarly, in $Path_1$, node β was chosen with probability $\frac{leaves_subtree_\beta}{leaves_subtree_r}$.

Now, note that the remaining portion of $Path_0$ is walking through a path in a B+Tree height four, whose root is α . Therefore, according to Algorithm 2, every path in this sub-tree has a uniform probability of being chosen (as was proved above). Hence, since this sub-tree has $leaves_subtree_\alpha$ leaves, then, every path has a probability of $\frac{1}{leaves_subtree_\alpha}$ for being chosen.

Equivalently, in the B+Tree height four whose root is β , every root-to-leaf path has a similar uniform probability of being chosen, which is $\frac{1}{leaves_subtree_\beta}$.

Hence, the probabilities of selecting these paths are as follows:

$$P(Path_0) = \frac{leaves_subtree_\alpha}{leaves_subtree_r} * \frac{1}{leaves_subtree_\alpha} = \frac{1}{leaves_subtree_r}$$

$$P(Path_1) = \frac{leaves_subtree_\beta}{leaves_subtree_r} * \frac{1}{leaves_subtree_\beta} = \frac{1}{leaves_subtree_r}$$

And we derive that indeed $P(Path_0) = P(Path_1)$.

Induction Hypothesis: Assume that the claim holds for height = k . It means that using the algorithm, all the leaves have an equal probability of being chosen. All that is left is to prove for height = $k + 1$.

Let $Path_0$ and $Path_1$ be two root-to-leaf paths, generated by the algorithm, in a given B+Tree of height $k + 1$, whose root is r . As in the base case, we will use α and β to denote the first nodes in the paths $Path_1$ and $Path_2$, respectively. According to the algorithm, the probability of choosing node α in $Path_0$ is $\frac{leaves_subtree_\alpha}{leaves_subtree_r}$. Similarly, the probability of choosing node β in $Path_1$ is $\frac{leaves_subtree_\beta}{leaves_subtree_r}$. Based on our hypothesis, in the B+Tree of height k whose root is α , all the leaves have an equal probability of $\frac{1}{leaves_subtree_\alpha}$ of being chosen. Similarly, in the B+Tree whose root is β , all the leaves have an equal probability of $\frac{1}{leaves_subtree_\beta}$ of being chosen. Therefore, we can conclude that:

$$\begin{aligned} P(Path_0) &= \frac{leaves_subtree_\alpha}{leaves_subtree_r} * \frac{1}{leaves_subtree_\alpha} = \frac{1}{leaves_subtree_r} \\ P(Path_1) &= \frac{leaves_subtree_\beta}{leaves_subtree_r} * \frac{1}{leaves_subtree_\beta} = \frac{1}{leaves_subtree_r} \end{aligned}$$

And we derive that indeed $P(Path_0) = P(Path_1)$ for any two paths in the $B + Tree$ of height $k + 1$, thus proving the generalized algorithm.

5 Simulation Study

In this section, we will use extensive simulations and demonstrate the superior performance of Algorithms 1 and 2 compared to the previous A/R-based methods while not affecting the sample quality.

5.1 Experiments Framework

In this section, we will use a simulation study to compare our proposed algorithms and Olken's [Olken and Rotem (1989)] over diverse settings of B+Tree heights (of three and four), data sizes, and sampling rates (of 0.5%, 1%, 5% and 10%). In order to examine the quality of the proposed algorithm in a complicated and realistic scenario, we used Zipfian distributed data [Poosala (1995)], which is considered to be a good representation of real-life applications [Makawita et al. (2002), Kluckhohn (1950)].

In all the figures in this section, we use the following notations: H for B+Tree height, Z for Zipfian skew factor, and D for domain size. The B+Tree was always of an order of 250, i.e., every node has at most $m = 250$ children. Also, our proposed algorithms are labeled as "distribution-oriented".

Random sampling algorithms are examined in terms of both their efficiency and the quality of randomness (i.e., goodness-of-fit). To verify the output samples are indeed following the full data distribution, we conducted a statistical

comparison based on Kolmogorov-Smirnov (KS) test as was done in previous related work [Shekelyan et al. (2022), Zhao et al. (2018)].

As previously stated in Sect. 3.3, we argue that practically, B+Trees of heights three and four are the most common cases, even for very large databases. In light of this, our series of experiments will examine the performance of the algorithms for B+Trees of these heights. Moreover, one can observe that the extensions for height four to the general case of any height are performed while maintaining uniform probability at the leaf level. Thus, the goodness-of-fit for the general case is anticipated to yield results similar to those obtained with a height of four. In the following simulations, we will also investigate the impact of conducting a uniform random walk up to the leaf level instead of the record level. It is pertinent to note that we have deliberately chosen not to include existing industry solutions for RDBMSs in our simulations (see Sect. 1) due to their inefficiency, specifically the requirement for full data scans, which results in an $O(n)$ complexity.

Furthermore, as evidenced by our proofs in Sect. 4, it is essential to highlight that each path has an equal probability of being selected. This probability remains independent of the B+Tree order parameter (m .) Consequently, the B+Tree order parameter remained consistent across all our experiments, as it is not anticipated to impact the sample’s quality or yield differing results regarding goodness-of-fit.

5.2 Experiments Setup

The experiments were conducted on two environments: MacBook Pro 2016 (2.0 GHz dual-core Intel Core i5 machine, with 32G RAM) and Google Colab (free Colab plan, CPU machine). All running-time experiments were performed on the same Google Colab environment.

5.3 Implementation Details

The B+Tree implementation were derived from the B+Trees open source³ (the “PURE_PYTHON” version). To conduct comprehensive experiments, we extended the Python library BTrees and implemented various sampling methods on it. For Olken’s algorithm, we implemented both the basic and early-abort versions. However, since both methods should provide similar results in terms of goodness-of-fit, we report here only the results of the early-abort version, which is the more efficient of the two. In addition to Olken’s algorithm, we implemented the BTWRS algorithm, as suggested in Makawita [Makawita et al. (2002)]. Overall, BTWRS performed similarly to Olken in goodness-of-fit. Still, its running time performance was significantly slower, implying impractical running times: 30s for sampling 10,000 values and up to 3h for sampling 20,000 values. Thus, we excluded it from the results section while publishing the entire code base online⁴.

³ <https://pypi.org/project/BTrees/>.

⁴ https://github.com/idancohen88/efficient_sampling.

5.4 Results

We followed Makawita [Makawita et al. (2002)] for generating the Zipfian data distribution and used different parameter values for the dataset size, domain size, and skew factor. We conducted experiments over a wide variety of settings, each illustrates a dataset with different characteristics. By changing these parameters, we could tune the number of distinct groups and their frequency in the dataset. Thus, the more skewed the data is, the more subsets appear with a low frequency, such that their likelihood of being sampled decreases.

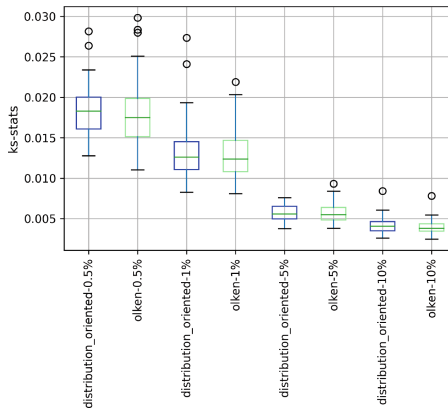


Fig. 3. Comparison of sampling results over Zipf’s data, using KS-tests for measuring goodness-of-fit, with $H = 3$ and the following pairs of (D, Z) parameters (1k, 0.2), (10K, 0.2), (10K, 0.5), (25K, 0.2), (25K, 0.5), (25k, 0.7). Each bar summarizes at least 60 experiments.

Figure 3 summarizes a series of experiments over B+Trees of height three (specifically, 1,000,000 values), with different Zipfian parameters, such that each specific experiment illustrates different dataset characteristics. The outcome of every experiment shows the same trend that we can clearly see in the graph: our algorithm provides the same performance as Olken’s in terms of goodness-of-fit. As expected, samples of 0.5% and 1% had a relatively high KS-score. Note that this inevitable outcome does not indicate a low algorithm quality.

Similarly, Fig. 4 summarizes experiments over B+Trees of height four and shows that the same trends and behavior also holds for the algorithm of height four. Also, in these experiments, Olken’s algorithm provided the same goodness-of-fit behavior as our algorithm. Both algorithms had the highest KS-score for a sampling of 0.5% of the data, with an average KS-score below 0.02. This score is reasonable given the small sample size of biased and skewed data. Accordingly, as the sample size increases, the score converges to a smaller value, around 0.005, which is practically negligible.

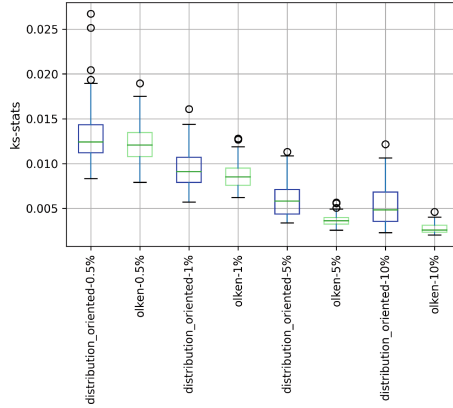


Fig. 4. Comparison of sampling results over Zipf’s data, using KS-tests for measuring goodness-of-fit, with all the possible settings with the parameters $H = 4$, $D = 10K$, $25K$, $500K$ and $Z = 0.2, 0.5, 0.7$. Each bar summarizes at least 60 experiments.

Running Times. As Olken’s algorithm and our algorithm are both stochastic algorithms, measuring the experiments’ running time is the most effective way to compare the algorithms’ performance.

Intuitively, the strength of our algorithms’ performance directly results from the fact that it does not imply even a single rejection. Thus, if the algorithm walks through a path and “touches” some nodes, these computer operations will always lead to a sampled value. This contrasts with Olken’s algorithm, where most paths will end with a rejection instead of a sampled value (as discussed in Sect. 1).

In Figs. 5(a) and 5(b), the box-plots show the running times, in seconds, of our algorithm (labeled as “distribution oriented”) and Olken’s algorithm. This is where our algorithms’ significant advantage comes into play, which does not use rejections and makes our algorithms run faster. For example, when sampling 0.5% of the data in a B+Tree of height three with 1,000 values, our algorithm finished in one second, while it took Olken’s early-abort 2.33s on average. Obviously, running times increase as the sampling size increases. Still, our algorithm stays in the range of a few seconds, while Olken’s reaches minutes: 20s to sample 100,000 with our algorithm, while Olken’s makes it in 454s (7 min) on average. Looking at bigger B-Trees and bigger sampling sizes, the differences become even more significant, and we can clearly see the outperformance of our proposed algorithms compared to Olken’s: On a B+Tree of 2 million values, our algorithm sampled 100,000 values in 45s on average, while Olken’s made it in 39 min.

Another compelling evidence for the differences between the algorithms’ results can be provided using Wilcoxon Rank Sum [Wilcoxon (1992)]: we compared the running times under each experiment’s settings (B+Tree size and sample size). We tested the null hypothesis that two sets of running times are drawn from the same distribution. All the experiments rejected the null hypothesis with a p-value below $0.1e^{-9}$.

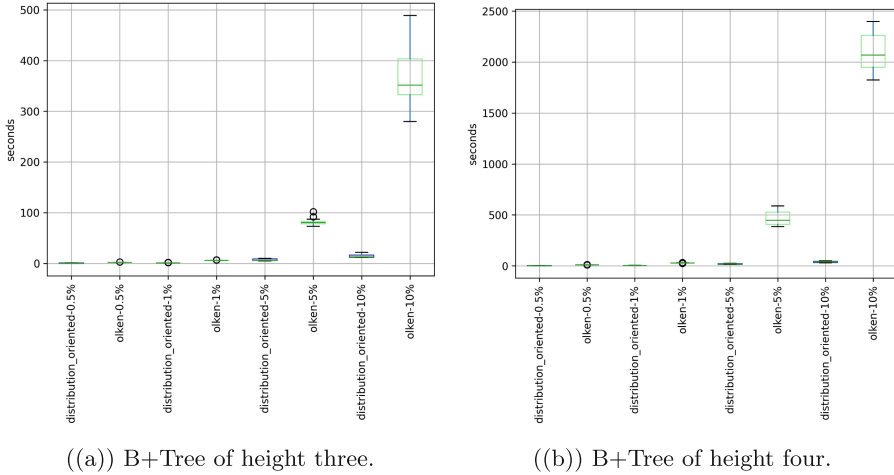


Fig. 5. Running times comparison. Each bar summarizes 25 experiments.

6 Summary and Future Work

In this paper, we studied the problem of random sampling in very large databases, which is a significant question in database research. We proposed a novel approach for random sampling over B+Trees, which performs an “intelligently-picked” random walks along the tree such that each time a leaf is visited, its value will be sampled without any rejections, unlike Olken’s and the other previous works, which resulted in a significant performance overhead. We proved the algorithms’ correctness and used an extensive simulation study to show the superior performance efficiency of our proposed algorithms compared to the existing state-of-the-art algorithms while not affecting the quality of the random sample.

Our future research will build upon this work to propose new state-of-the-art algorithms for efficient sampling in other database problems, such as sampling joins’ results, sampling with “where” conditions, parallel sampling, and sampling over distributed databases. We also plan to further study the task of random samples’ maintenance [Jermaine et al. (2004)].

References

- Abedjan, Z., Golab, L., Naumann, F.: Profiling relational data: a survey. *VLDB J.* **24**(4), 557–581 (2015)
- Agrawal, R., Kadadi, A., Dai, X., Andres, F.: Challenges and opportunities with big data visualization. In: *Proceedings of the 7th International Conference on Management of Computational and Collective intelligence in Digital EcoSystems*, pp. 169–173 (2015)

- Antoshenkov, G.: Random sampling from pseudo-ranked B+ trees. In: VLDB, pp. 375–382 (1992)
- Chaudhuri, S., Ding, B., Kandula, S.: Approximate query processing: no silver bullet. In: Proceedings of the 2017 ACM International Conference on Management of Data, pp. 511–519 (2017)
- Chaudhuri, S., Motwani, R., Narasayya, V.: Using random sampling for histogram construction. In: Proceedings of the ACM SIGMOD Conference, pp. 436–447 (1998)
- Comer, D.: Ubiquitous B-tree. *ACM Comput. Surv. (CSUR)* **11**(2), 121–137 (1979)
- Graefe, G., Kuno, H.: Modern B-tree techniques. In: 2011 IEEE 27th International Conference on Data Engineering, pp. 1370–1373. IEEE (2011)
- Haas, P.J.: Speeding up DB2 UDB using sampling. *IDUG Solut. J.* **10**(2), 6 (2003)
- Haas, P.J., Naughton, J.F., Swami, A.N.: On the relative cost of sampling for join selectivity estimation. In: Proceedings of the Thirteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 14–24 (1994)
- Hou, W.-C., Ozsoyoglu, G., Dogdu, E.: Error-constrained COUNT query evaluation in relational databases. *ACM SIGMOD Rec.* **20**(2), 278–287 (1991)
- Jermaine, C., Pol, A., Arumugam, S.: Online maintenance of very large random samples. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, pp. 299–310 (2004)
- Gluckhohn, C.: Human behavior and the principle of least effort (1950)
- Kudale, A.: B+ tree Preference over B Tree. Chicago, USA (n. d.). <http://www.academia.edu/11575258/B.tree.preference.over.B.trees>
- 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 (2016)
- Li, K., Li, G.: Approximate query processing: what is new and where to go? *Data Sci. Eng.* **3**(4), 379–397 (2018)
- Lipton, R.J., Naughton, J.F., Schneider, D.A., Seshadri, S.: Efficient sampling strategies for relational database operations. *Theor. Comput. Sci.* **116**(1), 195–226 (1993)
- Liu, Z., Zhang, A.: Sampling for big data profiling: a survey. *IEEE Access* **8**(2020), 72713–72726 (2020)
- Makawita, D., Tan, K.-L., Liu, H.: Sampling from databases using B+-trees. *Intell. Data Anal.* **6**(4), 359–377 (2002)
- Minkinen, P.: Practical applications of sampling theory. *Chemometr. Intell. Lab. Syst.* **74**(1), 85–94 (2004)
- Naughton, J.F., Seshadri, S.: On estimating the size of projections. In: Abiteboul, S., Kanellakis, P.C. (eds.) *ICDT 1990*. LNCS, vol. 470, pp. 499–513. Springer, Heidelberg (1990). https://doi.org/10.1007/3-540-53507-1_98
- Olken, F.: Random sampling from databases. Ph.D. Dissertation. University of California, Berkeley (1993)
- Olken, F., Rotem, D.: Random sampling from B+ trees. In: Proceedings of the 15th VLDB Conference, Amsterdam, The Netherlands (1989)
- Olken, F., Rotem, D.: Random sampling from databases: a survey. *Stat. Comput.* **5**(1), 25–42 (1995)
- Papaemmanouil, O., Diao, Y., Dimitriadou, K., Peng, L.: Interactive data exploration via machine learning models. *IEEE Data Eng. Bull.* **39**(4), 38–49 (2016)
- Piatetsky-Shapiro, G., Connell, C.: Accurate estimation of the number of tuples satisfying a condition. *ACM SIGMOD Rec.* **14**(2), 256–276 (1984)
- Poosala, V.: Zipf’s law (1995). citeseer.ist.psu.edu/116813.html
- Shekelyan, M., Cormode, G., Triantafillou, P., Shanghoosabad, A., Ma, Q.: Weighted random sampling over joins. arXiv preprint [arXiv:2201.02670](https://arxiv.org/abs/2201.02670) (2022)

- Slavakis, K., Giannakis, G.B., Mateos, G.: Modeling and optimization for big data analytics: (statistical) learning tools for our era of data deluge. *IEEE Signal Process. Mag.* **31**(5), 18–31 (2014)
- Vitter, J.S.: Random sampling with a reservoir. *ACM Trans. Math. Softw. (TOMS)* **11**(1), 37–57 (1985)
- Wilcoxon, F.: Individual comparisons by ranking methods. In: Kotz, S., Johnson, N.L. (eds.) *Breakthroughs in Statistics*. Springer Series in Statistics, pp. 196–202. Springer, New York (1992). https://doi.org/10.1007/978-1-4612-4380-9_16
- Wong, C.-K., Easton, M.C.: An efficient method for weighted sampling without replacement. *SIAM J. Comput.* **9**(1), 111–113 (1980)
- Wu, X., Zhu, X., Wu, G.-Q., Ding, W.: Data mining with big data. *IEEE Trans. Knowl. Data Eng.* **26**(1), 97–107 (2013)
- Zhao, Z., Christensen, R., Li, F., Hu, X., Yi, K.: Random sampling over joins revisited. In: *Proceedings of the 2018 International Conference on Management of Data*, pp. 1525–1539 (2018)



SQL-to-Schema Enhances Schema Linking in Text-to-SQL

Sun Yang^{1,3}, Qiong Su², Zhishuai Li³, Ziyue Li^{3(✉)}, Hangyu Mao³,
Chenxi Liu⁴, and Rui Zhao³

¹ Peking University, Beijing, China
2201210484@stu.pku.edu.cn

² Guizhou University, Guizhou, China
suqiong.gzu@gmail.com

³ SenseTime Research, Shanghai, China
{lizhishuai, liziyue, maohangyu, zhaorui}@sensetime.com

⁴ Nanyang Technological University, Singapore, Singapore
chenxi.liu@ntu.edu.sg

Abstract. Sophisticated Text-to-SQL methods often face errors, such as schema-linking errors, join errors, nested errors, and group-by errors. To mitigate these, it's crucial to filter out unnecessary tables and columns, focusing the language model on relevant ones. Previous methods have attempted to sort tables and columns based on relevance or directly identify necessary elements, but these approaches suffer from long training times, high costs with GPT-4 tokens, or poor schema linking performance. We propose a two-step schema linking method: first, generate an initial SQL query using the full database schema; then, extract the relevant tables and columns to form a concise schema. This method, tested with Code Llama and GPT-4, shows optimal performance compared to mainstream methods on the Spider dataset, reducing errors and improving efficiency in SQL generation.

Keywords: Text-to-SQL · Schema Linking · Large Language Model

1 Introduction

Text-to-SQL, a system that translates natural language queries into the underlying database language, enabling broader data access and usability [1]. The cross-domain dataset for Text-to-SQL, Spider [2], followed suit. The system needs to search databases to better understand what the user is querying [3]. Therefore, schema linking emerged. Schema linking is a specialized form of entity linking that associates phrases in a given question with column or table names in the database schema. SLSQL [4] illustrates the crucial role of schema linking in enhancing SQL parsing performance. Through relation-aware self-attention, RAT-SQL [5] introduces a unified framework to tackle schema encoding and linking challenges, learning schema and question representations jointly based

on their alignment and schema relations. SemQL [6] presents a neural approach for complex and cross-domain Text-to-SQL, aiming to address the lexical problem and the mismatch problem with schema linking and intermediate representation. Various fine-tuned [7] and prompting [8] Text-to-SQL methods with schema linking module appear spontaneously. However, these methods suffer from high training time costs and token consumption issues. We propose a novel approach: utilizing the complete schema and the question to compose a prompt for large language models to generate an initial SQL query, subsequently, parsing the initial SQL to extract columns and tables to form the linking schema. In summary, our contributions are:

1. We are the first to propose extracting the linking schema from the initial SQL and to define evaluation metrics for the schema linking module, allowing rapid validation of its effectiveness without waiting for SQL generation.
2. Combining our schema linking with our complete Text-to-SQL approach, using GPT-4 outperforms all zero-shot and few-shot prompt approaches, demonstrating the significant benefits of our schema linking for the overall Text-to-SQL task.

2 Related Work

2.1 Customized Machine Learning Fine-Tuned Methods

The traditional machine learning methods entail the concatenation of questions and database schemas into embeddings, training models to obtain softmax probabilities of relevance between these questions and columns. The tables and columns with the highest probabilities are selected as linking schemas. Such as, RESDSQL [7] selects the top 4 tables with the highest relevance. These approaches encounter challenges such as high training time cost, and diminished quality when applied to cross-domain datasets.

2.2 Stimulating General LLM with Prompting

The prompting method is mainly divided into two types: Few-shot prompting, such as DIN-SQL [8], provides multiple examples within the prompt, it leverages the large model’s in-context learning ability to enable the model to identify the precise schema. Zero-shot prompting, such as C3 [9], sorts the correlation between tables and columns from the database schema and the words in the question, and then selects the most relevant tables and columns as linking schema. These current prompting methods require the use of multiple complex modules with long prompts, which incur a significant amount of expensive GPT-4 token costs.

3 Methodology

We seek to enhance Text-to-SQL by harnessing the power of an initial SQL query, namely, SQL-to-Schema. Initially, we generate a preliminary SQL using the complete database schema and then employ SQL parsing methods to extract tables and columns, forming our linking schema. The GitHub address for our project is: <https://github.com/peking2025ys/SQL-to-Schema>.

3.1 Evaluation Metrics

To assess the quality of linking schemas, we consider that a schema must include all necessary tables and columns to generate the correct SQL. Each question in the Spider dataset involves up to 4 tables. We define `table-recall@4`, where a linking schema is considered correct if it includes all tables involved in the gold SQL. Otherwise, it is incorrect. Since the gold SQL and DIN-SQL schema include columns from aggregate functions like `count(*)`, and the C3 and RES-DSQL schemas do not, comparing columns may be unfair. Therefore, we use `table-recall@4` as one of the evaluation metrics in our experiments. Evaluating the quality of schema linking using only `table-recall@4` is insufficient. To address this, we introduce the SQL generation module, to combine the schemas obtained from different methods with the SQL generation module. We compare the quality of different schema linking schemes using the generated SQL. We employ the Text-to-SQL evaluation method mentioned in the Spider dataset, namely, execution accuracy [11], and supplement it as an additional evaluation metric for subsequent experiments.

3.2 Introduction to Each Module

Initial SQL Generation. Numerous studies have investigated prompting strategies for the Text-to-SQL task, conducting comprehensive comparisons of various prompt construction strategies for databases and demonstrations across zero-shot, single-domain, and cross-domain Text-to-SQL scenarios, such as, Din-SQL [8], C3 [9], Dail-SQL [10]. Leveraging the insights from these prompting research efforts, we ultimately designed the Initial SQL Generation(ISG) prompt illustrated in our Github page.

SQL Parse. This module is employed to extract the tables and columns labels from the gold SQL, simultaneously used to extract columns and tables from the SQLs of our other modules to form linking schemas.

SQL Generation. In order to further leverage the power of SQL-to-Schema, we iteratively utilize the SQL generation and parsing modules multiple times to enhance the quality of linking schemas and final SQLs.

Self-Consistency Voting. We conduct a vote among all generated SQL queries for the same question, selecting the most consistent SQL as the final output for Text-to-SQL. Our complete algorithm framework is illustrated in Fig. 1.

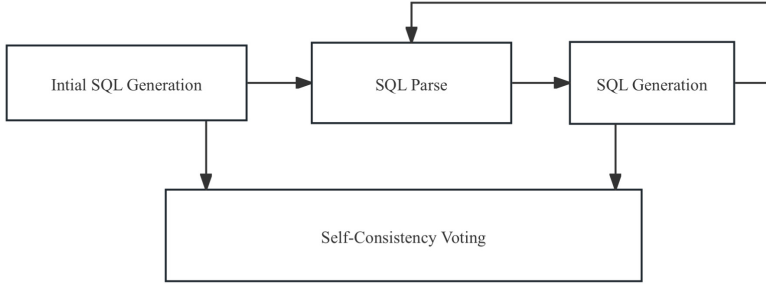


Fig. 1. The complete schema linking and Text-to-SQL algorithm framework.

4 Experiments and Analysis

4.1 Experiment One

When replicating the schema linking methods of DIN-SQL and C3 and implementing our method with Code Llama-34B, and incorporating schemas obtained from the RESDSQL method for comparison using table-recall@4, Our method achieved state-of-the-art performance in this metric (Table 1).

Table 1. Table-recall@4 for different schemas

| Schema | recall@4 | Schema | recall@4 |
|---------|----------|--------|----------|
| DIN-SQL | 0.88 | C3 | 0.93 |
| RESDSQL | 0.94 | Ours | 0.98 |

4.2 Experiment Two

To further validate the feasibility of SQL-to-Schema, the aforementioned linking schemas were input into the SQL generation module. Using Code Llama-34B, and the execution accuracy of SQLs was employed as a measure to represent the quality of schema linking. The experimental results are presented in Table 2. We observed that when all columns of a table in the linking schema appear in the SELECT clause of the gold SQL, the generated final SQL tends to use “SELECT *” which may output columns in that table that are unrelated to the question. Additionally, some linking schemas lack necessary tables and columns, resulting in errors in the generated SQL. Examples of these errors are illustrated in Our Github page. Therefore, in subsequent SQL generation, we use the linking schema as a reference while including the complete database schema in the prompt. Under the same experimental conditions except for the differing linking schemas, Table 2 demonstrates that the SQL-to-Schema strategy extracts higher-quality linking schemas.

Table 2. The results of SQL Execution Accuracy on the Spider dev dataset

| Approaches | EA | Approaches | EA |
|-------------------------------|-------|------------------------------|-------|
| DIN-SQL schema+SQL Generation | 0.723 | C3 schema+SQL Generation | 0.730 |
| RESDSQL schema+SQL Generation | 0.732 | SQL-to-Schema+SQL Generation | 0.753 |

4.3 Experiment Three

Since the schema linking module ultimately serves Text-to-SQL research task in both the NLP and DB communities, we cannot solely evaluate the quality of linking schemas and the feasibility of SQL-to-Schema from the perspective of the schema linking module. We must also consider whether the combination of our schema linking module and other functional modules related to Text-to-SQL tasks can bring benefits, namely, whether the gains from schema linking can be positively transferred to SQL generation. Therefore, we utilize GPT-4 turbo to run all modules, generating SQLs, and compare them with the currently best-performing fine-tuned models and zero shot and few shots prompting Text-to-SQL methods. The omarparison of Text-to-SQL experiments is shown in Table 3.

Table 3. The Comparison of Execution Accuracy in Text-to-SQL Methods

| Approaches | fine-tuned/prompting | EA |
|-------------------------|----------------------|-------|
| RESDSQL | fine-tuned | 0.841 |
| Dail-SQL | fine-tuned + fewshot | 0.824 |
| DIN-SQL | fewshot | 0.742 |
| C3 | zeroshot | 0.818 |
| SCVSQL+GPT4-turbo(ours) | zeroshot | 0.824 |

When employing the SQL-to-Schema method twice, namely, SCVSQL+GPT4-turbo, our approach surpassed all current zero shot and few shots prompting methods. The Dail-SQL method undergoes model training to select shot examples, and its few-shot setting consumes a substantial number of expensive GPT-4 tokens. In contrast, our method achieves comparable performance with minimal token consumption, showcasing the exceptional efficiency of our Text-to-SQL approach. While our method is only 0.019 lower than the optimal fine-tuned method, the RESDSQL method requires days of training time and significant GPU resources, whereas our approach only needs a multi-threaded CPU, completing all 1034 questions in the Spider Dev dataset within 10 min.

In order to further explore the upper limits of language models, for a specific problem, using our methods under the same language model, we conducted experiments, which employed the SQL-to-Schema module 0–3 times respectively. If a language model can correctly answer a specific question with the appropriate strategy or prompt, we believe that it has the capability to solve that ques-

tion. We recorded the maximum detection count of correct answers for the four models. We summarize various models alongside their respective upper limits, with Code Llama-34B achieving 0.8288, GPT4 reaching 0.8559, and GPT4-turbo attaining 0.8665. This tells us that language models heavily depend on prompts, and all current prompting methods have not yet reached the upper limit of the language model’s capabilities. There is still significant room for improvement in the use of prompting methods in the Text-to-SQL domain.

5 Conclusion

We introduced the SQL-to-Schema method for the first time by defining table-recall@4 and demonstrated the efficiency of this schema linking method on large Language models. Combined with the SQL generation module, we outperformed all prompting methods in Text-to-SQL tasks. This suggests that the approach of extracting schema using initial SQL can bring global benefits to the Text-to-SQL task, extending beyond the schema linking phase. It further confirms the necessity of schema linking methods in Text-to-SQL tasks. Additionally, our SQL generation performance indicates that when leveraging large language models, there is no need for complex modules typically used in traditional machine learning fine-tuned methods. Simple strategies suffice for achieving excellent results. The full potential of large Language models has not yet been fully explored, and there is still room for exploration in schema linking or Text-to-SQL tasks.

References

1. Katsogiannis-Meimarakis, G., Koutrika, G.: A survey on deep learning approaches for Text-to-SQL. *VLDB J.* **32**(4), 905–936 (2023)
2. Yu, T., et al.: Spider: a large-scale human-labeled dataset for semantic parsing and Text-to-SQL task. In: *Proceedings of EMNLP, Brussels, Belgium, Oct 31 - Nov 4*, pp. 3911–3921 (2018)
3. Yu, T., et al.: TypeSQL: knowledge-based neural Text-to-SQL generation. In: *Proceedings of NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, Vol. 2 (Short Papers)*, pp. 588–594 (2018)
4. Lei, W., et al.: Re-evaluating schema linking in Text-to-SQL. In: *Proceedings of EMNLP 2020*, pp. 6943–6954 (2020)
5. Wang, B., et al.: RAT-SQL: relation-aware schema encoding for Text-to-SQL parsers. In: *Proceedings of ACL 2020*, pp. 7567–7578 (2020)
6. Guo, J., et al.: Towards complex Text-to-SQL with intermediate representation. In: *Proceedings of ACL 2019, Florence, Italy, Jul 28 - Aug 2, Vol. 1: Long Papers*, pp. 4524–4535 (2019)
7. Li, H., et al.: RESDSQL: decoupling schema linking and parsing for Text-to-SQL. In: *Proceedings of 37th AAAI Conference on Artificial Intelligence*, pp. 13067–13075 (2023)
8. Pourreza, M., Rafiei, D.: DIN-SQL: decomposed in-context learning of Text-to-SQL with self-correction. *CoRR* [arXiv:2304.11015](https://arxiv.org/abs/2304.11015) (2023)
9. Dong, X., et al.: C3: zero-shot Text-to-SQL with ChatGPT. *CoRR* [arXiv:2307.07306](https://arxiv.org/abs/2307.07306) (2023)

10. Gao, D., et al.: Text-to-SQL empowered by large language models: a benchmark evaluation. CoRR [arXiv:2308.15363](https://arxiv.org/abs/2308.15363) (2023)
11. Zhong, R., Yu, T., Klein, D.: Semantic evaluation for Text-to-SQL with distilled test suites. In: Proceedings of EMNLP 2020, pp. 396–411 (2020)



Efficient Algorithms for Top-k Stabbing Queries on Weighted Interval Data

Daichi Amagata^(✉), Junya Yamada, Yuchen Ji, and Takahiro Hara

Osaka University, Suita, Osaka, Japan

{amagata.daichi,yamada.junya,ji.yuchen,hara}@ist.osaka-u.ac.jp

Abstract. This paper addresses the problem of processing top-k weighted stabbing queries on interval data. A state-of-the-art algorithm for this problem incurs $O(n \log k)$ time, where n is the number of intervals, so it is not scalable to large n . We solve this inefficiency issue and propose an algorithm that runs in $O(\sqrt{n} \log n + k)$ time. Furthermore, we propose an $O(\log n + k)$ algorithm to further accelerate the search efficiency.

Keywords: Interval data · Stabbing · Top-k query

1 Introduction

Many applications deal with interval data, where an interval is a pair of left and right endpoints. For example, objects associated with time information (e.g., sales items and vehicles) are usually maintained in interval format (e.g., the left and right endpoints are activation and termination time, respectively [2, 4–6]). In cryptocurrency and stock applications, the prices of cryptocurrencies and stocks vary continuously, and they record minimum and maximum prices (i.e., an interval) every certain time [10]. It is also intuitively known that each interval usually has a weight [1]. For instance, in the sales items and vehicles examples, the weights can be profits and the number of passengers, respectively.

To analyze the above weighted interval data, the following example query can be considered: Show k vehicles (e.g., trains) with the largest number of passengers at noon yesterday. This query helps consider a train operation plan and analyze train usage patterns for some events that occurred at a certain time. Motivated by such an application and usefulness, we address the problem of processing top-k weighted stabbing queries on interval data. Because a simple stabbing query does not consider weights and returns all stabbed intervals, applications cannot control the result size. That is, they may be overwhelmed by large result sizes, so the controllable result size (i.e., the top-k factor) is useful for such applications.

Given a set X of n weighted intervals and a query $q = (s, k)$ where s and k are respectively a query value and a result size, this query retrieves k intervals

stabbed by s with the largest¹ weight among X . An interval $x \in X$ is stabbed by q iff $s \in [x.l, x.r]$, where $x.l$ and $x.r$ are the left and right endpoints, respectively. Because many applications deal with large sets of intervals (i.e., n is large), an efficient algorithm for this problem is required. However, designing such an algorithm is non-trivial and challenging.

The most straightforward algorithm is as follows. We sort the intervals $\in X$ in descending order of weight offline. Given a top-k weighted stabbing query, we run a sequential scan of X until we access k stabbed intervals. Due to the sort order, (i) this set of the k intervals is guaranteed to be the exact top-k result, and (ii) this algorithm can stop the scan before accessing n intervals. However, in the worst case, this algorithm needs to access all intervals, so it incurs $O(n \log k)$ time. (The factor of $O(\log k)$ is required to update the intermediate top-k result.) Another approach is to employ a state-of-the-art algorithm [9]. This algorithm uses an interval tree [8] to find all stabbed intervals, and the top-k intervals are found from them. Because the interval tree structure guarantees that a (non top-k weighted) stabbing query can run in $O(\log n + m)$ time, where m is the number of stabbed intervals, this algorithm can run in $O(\log n + m \log k)$ for our problem. At first glance, this algorithm seems sufficiently fast, but it is important to notice that m can be as large as n (e.g., all intervals are stabbed by a query). Therefore, this algorithm results in the same worst time as the sequential scan.

We hence have a question: For our problem, does there exist an exact algorithm with less than $O(n)$ query time (and with $\tilde{O}(n)$ space, where $\tilde{O}(\cdot)$ hides any polylog factors)? We provide a positive answer and make the following contributions:

- An $O(\sqrt{n} \log n + k)$ time algorithm (Sect. 3). We first propose an algorithm that exploits weight-based sorting and the interval tree structure. This technique provides a performance guarantee dominating that of the state-of-the-art algorithm [9], because our algorithm runs faster than the state-of-the-art with the same space requirement. As $\sqrt{n} \log n < n$, we have $O(\sqrt{n} \log n + k) < O(n)$.
- An $O(\log n + k)$ time algorithm (Sect. 4). The second algorithm improves the search efficiency by exploiting the segment tree structure [7]. A segment tree yields the same performance for simple stabbing queries, i.e., its time complexity is $O(\log n + m)$, so simply applying this structure still incurs $O(n \log k)$ time in the worst case. Nevertheless, we show that a simple modification of this structure provides an $O(k \log n)$ time algorithm for our problem. We furthermore extend the segment tree to reduce the time complexity from $O(k \log n)$ to $O(\log n + k)$.
- Experiments on real datasets. We conduct experiments on two real large datasets. Due to space limitation, their results appear in [3].

2 Preliminary

Problem Definition. We use X to denote a set of n intervals. Each interval $x \in X$ is a pair of its left and right endpoints, i.e., $x = [x.l, x.r]$, where $x.l \leq x.r$.

¹ Some applications may prefer smaller weights, and our algorithms can deal with this case.

In addition, each interval $x \in X$ has an application-dependent static weight $w(x)$. Given a query value s , we say that x is stabbed by s iff $x.l \leq s \leq x.r$. For ease of presentation, we first define the stabbing query:

Definition 1 (Stabbing query). *Given a stabbing query s (which is a value) and X , this query retrieves a subset X_s of X such that $X_s = \{x \mid x \in X, x.l \leq s \leq x.r\}$.*

This paper considers a variant of stabbing queries and addresses the problem defined below.

Definition 2 (Top- k weighted stabbing query). *Given a top- k weighted stabbing query $q = (s, k)$, where s and k respectively are a query value and a result size, and X , this query retrieves k intervals with the largest weights among X_s . (If $|X_s| < k$, all intervals in X_s are returned.) Ties are broken arbitrarily.*

The state-of-the-art algorithm [9] requires $O(\log n + m \log k)$ time, where $m = |X_s|$. Theoretically, m can be as large as n , so it requires $O(n \log k)$ time in the worst case.

Interval Tree and Segment Tree. We introduce the interval tree structure [8], a building block of our algorithm presented in Sect. 3. Due to space limitation, we introduce only its theoretical result.

Lemma 1. *An interval tree can be built in $O(n \log n)$ time, consumes $O(n)$ space, and processes a stabbing query in $O(\log n + m)$ time, where m is the number of stabbed intervals.*

We next introduce the segment tree structure [7], because we use it as a building block of our algorithm presented in Sect. 4. Again, we introduce only its theoretical result.

Lemma 2. *A segment tree can be built in $O(n \log n)$ time, consumes $O(n \log n)$ space, and processes a stabbing query in $O(\log n + m)$ time, where m is the number of stabbed intervals.*

3 Algorithm Based on Interval Forest

This section proves the following theorem. (Missing proofs appear in the full version of this paper [3].)

Theorem 1. *For our problem, there exists an exact algorithm that needs $O(n \log n)$ pre-processing time, $O(n)$ space, and $O(\sqrt{n} \log n + k)$ query time.*

Main Idea. The main idea of this algorithm is to combine weight-based sorting and the interval tree structure. Assume that the intervals in X are sorted in descending order of weight. Now assume that X is partitioned into two disjoint subsets X_1 and X_2 , and note that $w(x) \geq w(x')$ for all $x \in X_1$ and $x' \in X_2$. Next consider that two interval trees \mathcal{I}_1 and \mathcal{I}_2 are built, i.e., \mathcal{I}_1 (\mathcal{I}_2) is built on X_1 (X_2). Given a top-k weighted stabbing query, we first use \mathcal{I}_1 . If \mathcal{I}_1 returns k stabbed intervals, we do not need to use \mathcal{I}_2 , since the weights of the intervals in \mathcal{I}_2 are less than those of the intervals in \mathcal{I}_1 . Based on this observation, we reduce the $O(n \log k)$ time of [9] to $O(\sqrt{n} \log n + k)$.

3.1 Data Structure and Construction

We sort the intervals $\in X$ as above. Then, we partition X into p equal-sized disjoint subsets, i.e., $X = X_1 \cup X_2 \cup \dots \cup X_p$ and $X_i \cap X_j = \emptyset$ ($i \neq j$). In addition, $w(x) \geq w(x')$ for all $x \in X_i$, $x' \in X_{i+1}$ ($i \in [1, p-1]$). We later show how to specify p , which is an important factor for achieving a solid performance guarantee. Then, we build an interval tree for each subset of X , so we have p interval trees. Note that (i) this structure is general for arbitrary top-k weighted stabbing queries, meaning that this pre-processing is done only once, and (ii) Lemma 1 directly derives the following.

Corollary 1. *We can build p interval trees in $O(n \log n)$ time, and they require $O(n)$ space in total.*

3.2 Query Processing Algorithm

Our algorithm proposed in this section is denoted by IF because this algorithm employs multiple interval trees, i.e., Interval Forest. Given a top-k weighted stabbing query q , IF first uses the interval tree \mathcal{I}_1 on X_1 and runs q on \mathcal{I}_1 . IF uses the stabbing query processing algorithm on the interval tree structure to find stabbed intervals. Whenever IF accesses a stabbed interval, it updates the top-k result. If the number of stabbed intervals is equal to or more than k , it is guaranteed that we can obtain the exact top-k result from \mathcal{I}_1 , so IF returns the result. Otherwise, IF runs q on \mathcal{I}_2 , and IF repeats this iteration until we have k stabbed intervals or all interval trees are used.

Analysis. We set $p = O(\sqrt{n})$, so we have $O(\sqrt{n})$ interval trees and $|X_i| = O(\sqrt{n})$ for each $i \in [1, p]$. Then, we have:

Lemma 3. *IF runs in $O(\sqrt{n} \log n + k)$ time.*

Proof of Theorem 1. From Corollary 1 and Lemma 3. □

4 Algorithm Based on a Variant of Segment Tree

We next consider accelerating the search efficiency further (by sacrificing pre-processing time and the space complexity a bit) and prove that

Theorem 2. *For our problem, there exists an exact algorithm that requires $O(n \log n \log \log n)$ pre-processing time, $O(n \log^2 n)$ space, and $O(\log n + k)$ query time.*

Main Idea. This algorithm is designed based on the segment tree structure. One may come up with the idea of sorting the intervals maintained in each node of a segment tree based on weight. This idea enables access to at most k intervals for each traversed node. As the height of the segment tree is $O(\log n)$, this idea derives an $O(k \log n)$ time algorithm. Although this algorithm can theoretically be faster than IF, its running time can be sensitive to k . We therefore do not employ this approach.

Instead, we focus on the following property: the stabbing query algorithm on the segment tree structure exploits the fact that the intervals maintained in the traversed nodes are guaranteed to be stabbed by a given query. Then, by storing all intervals existing in the path from the root to each node in a sorted array, we do not need to enumerate k intervals for each traversed node². This new idea and the path-based auxiliary structure are specific to our problem, since simple stabbing queries enumerate all stabbed intervals and do not consider weights.

4.1 Variant of Segment Tree and Its Construction

We first build a segment tree on X . Then, for each node u of the segment tree, we consider the path from u_{root} to u . We collect all “distinct” intervals maintained in the nodes on the path (since duplicate intervals may exist in the path), and u stores this set of intervals in a weight-based sorted array.

After making this sorted array for each node u of the segment tree, we remove a set of intervals initially maintained in u because we do not use it anymore. Note that this structure is also general to arbitrary top- k weighted stabbing queries, so this pre-processing is done only once. We analyze this pre-processing time and the space complexity of this structure.

Lemma 4. *We can build the above variant of a segment tree in $O(n \log n \log \log n)$ time.*

Lemma 5. *The above variant of a segment tree needs $O(n \log^2 n)$ space.*

² This idea is not available for the interval tree structure. This is because the interval tree structure does not guarantee that all intervals maintained in a node are stabbed by a given query.

4.2 Query Processing Algorithm

Now we are ready to present our second algorithm for the top-k weighted stabbing queries. Thanks to our non-trivial extension of the segment tree structure, we can design a simple and fast algorithm. This algorithm is denoted by ST-PSA (Segment Tree with Path-based Sorted Arrays).

Let \mathcal{S} be our variant of a segment tree on X . Given a top-k weighted stabbing query $q = (s, k)$, ST-PSA first runs a simple stabbing query $q.s$ on \mathcal{S} and obtains the node traversed last during the stabbing. Let this node be u , and ST-PSA uses the sorted array of u . Specifically, ST-PSA returns the first k intervals in the array as the top-k result.

Correctness. The stabbing query algorithm on the segment tree structure guarantees that all intervals maintained in the traversed nodes are stabbed by a given query. In addition, the sorted array of u stores all intervals (initially) maintained in the path from u_{root} to u . From these facts, the correctness of ST-PSA is clear.

Time Complexity. We present the main result of this section below.

Lemma 6. *ST-PSA runs in $O(\log n + k)$ time.*

Proof of Theorem 2. From Lemmas 4–6. □

5 Conclusion

This paper addressed the problem of processing top-k weighted stabbing queries. A state-of-the-art algorithm for this problem incurs the same time complexity as that of a sequential scan. Motivated by this inefficiency issue, this paper proposed two sublinear time algorithms.

Acknowledgements. This work was partially supported by AIP Acceleration Research JPMJCR23U2, JST, and JSPS KAKENHI Grant Number 24K14961.

References

1. Agarwal, P.K., Arge, L., Yi, K.: An optimal dynamic interval stabbing-max data structure? In: SODA, pp. 803–812 (2005)
2. Amagata, D.: Independent range sampling on interval data. In: ICDE (2024)
3. Amagata, D., Yamada, J., Ji, Y., Hara, T.: Efficient algorithms for top-k stabbing queries on weighted interval data (full version). [arXiv:2405.05601](https://arxiv.org/abs/2405.05601) (2024)
4. Behrend, A., et al.: Period index: a learned 2D hash index for range and duration queries. In: SSTD, pp. 100–109 (2019)
5. Christodoulou, G., Bouros, P., Mamoulis, N.: HINT: a hierarchical index for intervals in main memory. In: SIGMOD, pp. 1257–1270 (2022)

6. Christodoulou, G., Bouros, P., Mamoulis, N.: HINT: a hierarchical interval index for allen relationships. *VLDB J.* **33**, 73–100 (2023). <https://doi.org/10.1007/s00778-023-00798-w>
7. De Berg, M.: *Computational Geometry: Algorithms and Applications* (2000). <https://doi.org/10.1007/978-3-540-77974-2>
8. Edelsbrunner, H.: *Dynamic Rectangle Intersection Searching* (1980)
9. Xu, J., Lu, H.: Efficiently answer top-k queries on typed intervals. *Inf. Syst.* **71**, 164–181 (2017)
10. Zhang, Z., Gan, J., Bao, Z., Kazemi, S.M.H., Chen, G., Zhu, F.: Approximate range thresholding. In: *SIGMOD*, pp. 1108–1121 (2022)



A Hierarchical Storage Mechanism for Hot and Cold Data Based on Temperature Model

Shicong Ma, Tianhai Zhao, Jianhua Gu, and Yunlan Wang^(✉)

Northwestern Polytechnical University, Xi'an, Shaanxi, China
masc@mail.nwpu.edu.cn, {zhaoth, gujh, wangyl}@nwpu.edu.cn

Abstract. The storage of hot and cold data play a crucial role in improving data access efficiency and reducing storage expenses. This paper proposed a temperature model to quantify the real-time hotness of data. Based on the temperature model, we proposed a hierarchical storage mechanism for hot and cold data, managing dynamic data migration among local cold database, local hot database, and remote cold database. Experimental results show the advantages of the proposed method in terms of hot data hit rate, hot data hit rate for key data, migration count, and average response time. It can improve data access performance and the satisfaction of important users, and significantly reduce expenses.

Keywords: hot and cold data · hierarchical storage · data temperature model · dynamic migration · remote storage

1 Introduction

In recent years, the data volume expanded rapidly and the demand for data management and processing capabilities has steadily increased, posing unprecedented challenges in the storage and management of data. The Pareto Principle shows the majority of accesses concentrate on a small portion of data over a specific period [1], enabling us to allocate data with varying hotness to distinct storage media for hierarchical storage. Commonly used storage devices include Hard Disk Drive (HDD) and Solid State Drive (SSD). HDD provide economical and substantial storage capacity solutions. Conversely, SSD offer faster read and write speed, but come with higher price. As urban land development approaches saturation in many developed cities, essential resources like water and electricity face strain. The concepts of channel more computing and storage resources from developed regions to less developed but resource rich regions have gained considerable attention and development in country like China [2]. An effective hierarchical storage strategy can optimize data access performance while conserving storage expenses, but it also imposes higher requirements for accurately identifying the degree of data hotness.

This article proposes a **hierarchical storage mechanism** for hot and cold data based on **temperature model** (HSTM). Firstly, we construct the data temperature model. Then, based on the temperature model, we propose hierarchical storage mechanism. The main contributions of HSTM are:

- **Data temperature model.** Propose a model for computing temperature based on access time, access frequency, user priority, and data priority. This model provides a comprehensive assessment and quantification of the real-time hotness of data.
- **LSTM network based prediction.** Employ Long Short Term Memory (LSTM) networks to forecast the volume of new data, striking a balance between the utilization of hot database space and data migration count.
- **Periodic migration pattern analysis.** Mine the periodic migration pattern of data. Pre-migration of data with identifiable periodic migration patterns can mitigate time latency of migration, further enhancing the hot data hit rate.
- **Cost-based decision for local and remote storage.** When the local cold database storage space is insufficient, migrate the data with higher benefits of storing remotely to the remote cold database, substantially reducing storage expenses.

2 Related Work

The predominant methods for identifying hot and cold data encompass two classical cache replacement algorithms: the Least Recently Used algorithm (LRU) [3] and the Least Frequently Used algorithm (LFU) [4]. Various optimizations and variations, such as LRU-K [5] and LFU-Aging [6], have been proposed to enhance the adaptability to different scenarios.

In subsequent investigations, scholars have adopted intricate data structures to discern between hot and cold data. HashKV proposes a distinction strategy between hot keys and cold keys [7]. HashKV stores the hot keys in the segment of vLog, and separates the cold key, stores in the disk then. HotRing proposed an innovative Key-Value Store (KVS) with hotspot awareness, specifically designed for massively concurrent access to a small portion of items [8]. Facebook identified hot and cold data based on the status of a bucket of objects instead of the status of a single object [9].

Some algorithms have achieved commendable performance, extending beyond the realm of data structures. Xie compared the data's access frequency with a pre-defined threshold [10], then determine whether the data is hot or cold.

Recent years have introduced approaches that assess data hotness through the evaluation of data temperature values. Song have devised a temperature calculation model that exhibits sensitivity to temporal changes, drawing upon Newton's Law of Cooling [11].

Most current methods designed for the identification and management of hot and cold data rely on basic access characteristics, such as access frequency and

time. However, these methods often neglect the opportunity to enhance important users' satisfaction and do not optimize methods from the perspective of mining data access and migration patterns. Meanwhile, there is little consideration given to the performance and storage cost of local and remote storage.

3 System Architecture

HSTM comprises two components: the data temperature model and the hierarchical storage mechanism for hot and cold data.

The initial section quantifies the real-time hotness of data. In this paper, data stored in the hot database is categorized as hot data, while data stored in the cold database is categorized as cold data. The subsequent section, built upon the temperature model, manages the data dynamic migration.

3.1 Data Temperature Model

The new data are likely to access in the immediate future and thus warrants storage in the hot database. The initial temperature value $T_{newData}$ is set as the average temperature of hot data.

Using I and Q to represent the user and data priority, where $I = 1$ indicates an important user, $I = 0$ indicates a regular user, $Q = 1$ indicates key data, and $Q = 0$ indicates regular data. Let T_{pre} and T_{new} denote the temperature before and after data access, T_{new} can be formulated as follows:

$$T_{new} = T_{pre} + \alpha + \beta \times I + \gamma \times Q \quad (1)$$

where α , β , and γ denote weights of access, user priority, and data priority.

In order to mitigate the persistent impact of short-term frequent access on data temperature, it is necessary to periodically reduce the temperature of data. Let T_{pre} and T_{new} be the data temperature before and after reduction, and δ be the temperature weakening coefficient. T_{new} is the product of T_{pre} and δ .

3.2 Data Hierarchical Storage Mechanism

Local Hot and Cold Data Migration Management includes time-driven and event-driven migration. Time-driven migration occurs at the end of each period p , event-driven migration occurs in the process of each period p .

To mine the periodic migration pattern of data, when data d is migrated from cold to hot database for the y -th time (where $y \geq 4$), if the three intervals resulting from the last four migrations are equal (the difference in period numbers between two consecutive migrations form interval), it is inferred that data d exhibits a periodic migration pattern.

In time-driven migration, we utilize LSTM to predict new data volume of upcoming period e based on new data volume of previous period, then compute the upper threshold of the current period ζ as the difference between the hot

database storage capacity c and 0.5 times e , indicating the maximum storage space utilized in the hot database post-migration. By means of migration, the cold data predicted to migrate to the hot database in the next period, along with a portion of the currently high-temperature data, is stored in the hot database.

Event-driven migration involves storing newly generated data and cold data with increased temperature in the hot database. If the storage space in the hot database is inadequate, hot data is evicted to the local cold database in ascending order of temperature until sufficient storage space is adequate.

Local and Remote Data Migration Management includes two types of event-driven migration. When the temperature of remote cold data increases, it initiates the migration of remote cold data back to the local database. When the utilized storage space of the local cold database exceeds 90% of the total storage capacity, it initiates the migration of local cold data to the remote database.

We denote the benefit of storing a unit of data remotely as udy , can be expressed as the difference between the local storage cost $cost_L$ and the remote storage cost $cost_R$, divided by the data size s , and $cost$ is the weighted sum of the monetary cost $expCost$ and the latency cost $dlyCost$. The $expCost$ is calculated as the product of the data size s , the power consumption per unit of stored data e , and the unit cost of electricity ep . The $dlyCost$ is the sum of the sending latency $s/bindwidth(bw)$ and the propagation latency pd , multiplied by the normalized data temperature T' .

$$udy = \frac{cost_L - cost_R}{s} = \frac{s \times e \times epdiff + \theta \times (\frac{s}{bw_L} - \frac{s}{bw_R} - pd_R) \times T'}{s} \quad (2)$$

where θ is the weight of the delay cost relative to the monetary cost, and $epdiff$ represents the difference in electricity cost per unit between local and remote storage, and pd_L in $dlyCost_L$ is negligible.

4 Experimentation and Analysis

This section conducts simulation experiments to compare the HSTM with the LRU-K and LFU-Aging algorithms.

Evaluation metrics encompassed the hot data hit rate for all data, hot data hit rate for key data, data migration count, and average access response time.

4.1 Experimental Environment and Configuration

The experiment utilized two computers to emulate local and remote storage, each equipped with an Intel(R) Xeon(R) Gold 6132 CPU @ 2.60 GHz, 376 GB of memory, and CentOS Linux release 7.4.1708 (Core). One computer was configured with a 1.5 TB SSD and a 20 TB HDD, while the other had a 35 TB HDD.

An initial set of one million data files, ranging in size from 1 MB to 20 MB, were stored in the local database. Each period comprised around 1 TB hotspot

data, with 80% being the previous period’s hotspot data, 10% from the previous period’s non-hotspot data, and 10% newly generated data in the current period. Each data access had an 80% probability of targeting hotspot data. The period p was set to one day, conducting a total of 60 million accesses over 30 days. Periodic temperature reductions occurred after the completion of each time-driven migration.

The settings for parameters are as follows: the SSD read speed is 500 MB/s, HDD read speed is 100 MB/s, bandwidth of long-distance link is 1G, pd_R is 20 ms, ε is 4, α is 0.5, β is 0.25, γ is 0.25, δ is 0.5 and m is 8.

4.2 Experiment on Local Hot and Cold Data Migration Management

The value of c was set as 0.8 TB, 1 TB, and 1.2 TB. The results are presented below (Table 1).

Table 1. Experimental results of HSTM, LRU-K, and LFU-Aging.

| c | Metric | HSTM (=0.25/0.75/1.25) | LRU-10 | LRU-15 | LRU-20 | LFU-Aging |
|-----|-----------------------------------|---------------------------|--------|--------|--------|-----------|
| 0.8 | hit rate (%) | 59/59.3/59.3 | 55.5 | 56.7 | 57.2 | 53.6 |
| | hit rate for key data (%) | 72.7/75.8/76.5 | 65.8 | 66.6 | 66.7 | 68.8 |
| | migration count (millions) | 3.13/2.83/2.74 | 4.96 | 2.93 | 2.05 | 56.1 |
| | average access response time (ms) | 52.8/52.56/52.56 | 55.6 | 54.64 | 54.24 | 57.12 |
| 1 | hit rate (%) | 72.4/72.5/72.4 | 67.8 | 67.9 | 67.5 | 64.1 |
| | hit rate for key data (%) | 87.6/89.6/90 | 80.6 | 80.4 | 79.8 | 81 |
| | migration count (millions) | 1.87/1.64/1.56 | 3.54 | 2.09 | 1.48 | 43.56 |
| | average access response time (ms) | 42.08/42/42.08 | 45.76 | 45.68 | 46 | 48.72 |
| 1.2 | hit rate (%) | 77.6/77.5/77.2 | 75.3 | 74.2 | 72.2 | 71.7 |
| | hit rate for key data (%) | 93.1/93.3/93.4 | 89.9 | 88.4 | 85.7 | 88.6 |
| | migration count (millions) | 1.25/1.24/1.23 | 2.68 | 1.62 | 1.29 | 34.46 |
| | average access response time (ms) | 37.92/38/38.24 | 39.76 | 40.64 | 42.24 | 42.64 |

The experimental results show that HSTM exhibits an advantage in the hot data hit rate, the hot data hit rate for key data, and the average response time. When c surpasses the volume of hotspot data s , HSTM exhibits an advantage in migration count. Moreover, increasing the value of the data attribute weight parameter γ leads to a further improvement in the hot data hit rate for key data.

4.3 Experiment on Local and Remote Data Migration Management

The total storage capacities of the local cold database were set to 1 TB, 2 TB, and 3 TB, respectively. The parameter θ in (2) adjusted dynamically in each migration, ensuring that, subsequent to the data migration to the remote location, the utilized storage space in the local cold database accounted for approximately 60% of its total storage capacity.

The experimental results show that compared to storing all data locally, adopting local and remote hierarchical storage increases the average response time from 42.08 ms to 45.66 ms, 45.5 ms and 45.24 ms when the local cold data capacity is 1 TB, 2 TB and 3 TB, respectively. Assuming the volume of data stored remotely is v PB, the annual cost for the long-distance link is 200,000 yuan, and the power consumption rate for storing data e is 2.5 KW/PB. The price difference in electricity costs between local and remote locations $epdiff$ is 0.4 yuan per KWH. The formula for calculating the annual savings in storage costs $expSavings$ for local and remote hierarchical storage is expressed as follows:

$$expSavings = 2.5 \times 24 \times 365 \times 0.4 \times v - 200000 \quad (3)$$

5 Conclusion

This paper proposes a hierarchical storage mechanism for hot and cold data based on temperature model, implementing a three-tiered data storage and management encompassing local SSD, local HDD, and remote HDD, offering advantages in hot data hit rate and migration count, and reducing storage expenses. Future research should delve deeper into data access patterns in data centers to develop targeted data hierarchical storage strategies.

Acknowledgments. This work is supported by the National Key R&D Program of China (NO. 2022YFB4501701).

References

1. Sanders, R.: The Pareto principle: its use and abuse. *J. Serv. Mark.* **1**, 37–40 (1987)
2. Eastern Data, Western Computing: China’s National Big Data Infrastructure Project. rootaccess.substack.com/p/eastern-data-western-computing-chinas. Accessed 16 Mar 2024
3. Dan, A., Towsley, D.: An approximate analysis of the LRU and FIFO buffer replacement schemes. In: *Proceedings of the 1990 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pp. 143–152. ACM (1990)
4. Robinson, J.T., Devarakonda, M.V.: Data cache management using frequency-based replacement. In: *Proceedings of the 1990 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pp. 134–142. ACM (1990)
5. O’Neil, E.J., O’Neil, P.E., Weikum, G.: The LRU-K page replacement algorithm for database disk buffering. *ACM SIGMOD Rec.* **22**(2), 297–306 (1993)
6. Arlitt, M., Friedrich, R., Jin, T.: Performance evaluation of web proxy cache replacement policies. In: Puigjaner, R., Savino, N.N., Serra, B. (eds.) *TOOLS 1998*. LNCS, vol. 1469, pp. 193–206. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-68061-6_16
7. Chan, H.H., et al.: HashKV: enabling efficient updates in KV storage via hashing. In: *USENIX ATC 2018*, pp. 1007–1019 (2018)
8. Chen, J., et al.: HotRing: a hotspot-aware in-memory key-value store. In: *18th FAST*, Santa Clara, pp. 239–252 (2020)

9. Muralidhar, S., et al.: f4: Facebook's warm BLOB storage system. In: 11th OSDI, Broomfield, pp. 383–398 (2014)
10. Xie, Y., et al.: Efficient storage management for social network events based on clustering and hot/cold data classification. *IEEE Trans. Comput. Soc. Syst.* **10**, 120–130 (2023)
11. Song, Y., et al.: A novel hot-cold data identification mechanism based on multidimensional data. In: 2022 5th DSIT, Shanghai, pp. 1–5 (2022)

Machine Learning and Large Language Models



A Pre-trained Knowledge Tracing Model with Limited Data

Wenli Yue^{1,3}, Wei Su^{1,3}(✉), Lei Liu², Chuan Cai¹, Yongna Yuan¹,
Zhongfeng Jia¹, Jiamin Liu¹, and Wenjian Xie¹

¹ School of Information Science and Engineering, Lanzhou University,
Lanzhou, China
suwei@lzu.edu.cn

² Duzhe Publishing Group Co. Ltd., Lanzhou, China

³ Key Laboratory of Media Convergence Technology and Communication,
Lanzhou, Gansu, China

Abstract. Online education systems have gained increasing popularity due to their capability to fully preserve users' learning data. This advantage enables researchers to assess learners' mastery through their learning trajectories, thereby facilitating personalized education and support. Knowledge tracing, an effective educational aid, simulates students' implicit knowledge states and predicts their mastery over knowledge based on their historical answer records. However, for newly developed online learning platforms, the lack of sufficient historical answer data may impede accurate prediction of students' knowledge states, rendering existing knowledge tracing models less effective. This paper introduces the first pre-trained knowledge tracing model that leverages a substantial amount of existing data for pre-training and a smaller dataset for fine-tuning. Validated across several publicly available knowledge tracing datasets, our method demonstrates significant improvement in tracing performance on small datasets, with a maximum AUC increase of 5.07%. Beyond incorporating small datasets, our approach of pre-training the entire dataset has shown an enhanced AUC compared to the baseline, marking a novel direction in knowledge tracing research. Furthermore, the paper analyzed the outcomes of pre-training experiments with varying numbers of interactions as fine-tuning datasets, providing valuable insights for Intelligent Tutoring Systems (ITS).

Keywords: Knowledge Tracing · Limited Data · Pre-training · Fine-tuning

1 Introduction

In recent years, the spread of mobile applications has transformed educational strategies, with Intelligent Tutoring Systems (ITS) [2] and Massive Open Online Courses (MOOCs) [23] becoming increasingly prevalent. These platforms utilize data-driven techniques to personalize learning paths, enhancing student engagement and educational access. One pivotal technique is knowledge tracing, which predicts student mastery over time [1, 7].

Corbett et al. [5] introduced the Bayesian Knowledge Tracing (BKT) model, combining elements from Hidden Markov Models [3], and Bayesian networks [21] to model students' mastery of knowledge concepts as binary variables based on real-time feedback. The Deep Knowledge Tracing (DKT) [11] was the first to apply recurrent neural networks to knowledge tracing tasks. It uses LSTM models to track the dynamic changes in students' knowledge proficiency over time and directly learns the latent vector representation of students' proficiency from datasets. The task of knowledge tracing is to model students' knowledge mastery status based on their answer records, exercises, knowledge components (KCs), and students' answer, in order to predict their future answer. Despite DKT's progress, it fails to distinguish between different problems under the same knowledge concept, prompting developments of models that address these shortcomings.

To enhance the management of long-term dependencies within sequences, Sha et al. [13] proposed an improved model that employs a stacked LSTM network, which vertically stacks two LSTM networks and introduces a residual network to prevent the degradation of network performance. Yang et al. [24] introduced a decision-tree-based DKT that combines random forests and gradient boosting for better accuracy. Zhang et al. [29] used autoencoders for feature learning, and the qDKT model [17] focused on question-level input refinement. MANN [22] introduced memory matrices to RNNs, enhancing DKT's handling of extensive interaction data. DKVMN [28] is designed with a dynamic value matrix while retaining a static key matrix, allowing for a more precise tracking of the changes in students' knowledge states over time. Attention mechanisms have been integrated into knowledge tracing models like SAKT [9] and AKT [6] to address limitations in handling sparse data and limited interactions. Models such as SAINT [4], built on the Transformer [20] framework, and its extension SAINT+ [14], incorporate features like time spent on questions to enhance knowledge tracing. The RKT model [10] employs attention mechanisms to capture the decay in students' memory over time. Graph-based approaches like GKT [8] and GIKT [25] model knowledge tracing as a node classification problem over time, using graph algorithms to capture complex relationships between problems and skills. SKT [19] focuses on the multifaceted relationships between knowledge components, and HGKT [27] exploits hierarchical problem graphs to improve tracing. Bi-CLKT [16] introduces a novel contrastive learning framework for a deeper understanding of knowledge relationships.

Combating the "cold start" problem in smart education, using attention mechanisms with Neural Turing Machines (NTM) has been effective [30]. This approach, especially with small datasets, allows the Logic-Muse ITS to accurately track students' knowledge states by incorporating prior knowledge into DKT. However, its effectiveness in predicting outcomes on other publicly available datasets has not been deemed satisfactory [18].

These studies focus on a deep exploration of problems and skills, but accurately predicting students' knowledge states from limited educational data based on historical interactions (such as a few hundred or fewer) remains a challenging task.

The main contributions of this work are described as follows:

- This paper presents the first model that integrates knowledge tracing tasks with a “pre-training + fine-tuning” strategy. The method involves pre-training on data from various ITS and subsequently fine-tunes on a limited amount of interaction data from a distinct system than the pre-training data. The approach offers a promising solution to address the “cold start” challenge in ITS.
- A comparative analysis of the results from varying numbers of interactions used as fine-tuning datasets in pre-training experiments is conducted. Additionally, we assess the effectiveness of the pre-training method by employing a comprehensive knowledge tracking dataset. Our findings demonstrate that our proposed method outperforms alternative models.

2 Related Work

Pre-trained Models (PTMs) are models that have been pre-trained on a specific foundational task, with the aim of becoming a universal model within a certain domain by learning a complex hierarchy of features. After being pre-trained, these models can be retrained or fine-tuned on different but related target tasks, enhancing the performance of the target task by leveraging the parameters and knowledge learned from the original task, as illustrated in Fig. 1. In recent years, various PTMs have been proposed to better acquire knowledge from unlabeled data, with improvements in model architecture and pre-training tasks through models such as XLNet [26], and MASS [15]. Additionally, researchers have explored building large-scale models with billions of parameters, like the GPT [12] series, while optimizing the computational efficiency of training PTMs.

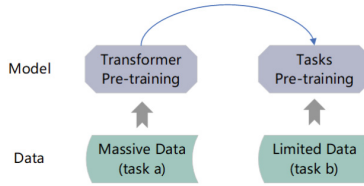


Fig. 1. Pre-trained model framework diagram

Pre-trained models often have a good semantic representation of features, and the portability of these learned features between different problems is a crucial advantage of deep learning compared to many early shallow learning methods. This advantage makes deep learning effective even for small data problems.

3 Method

To address the cold start problem, this study introduces a knowledge tracing model based on pre-training and fine-tuning techniques, named PMKT (Pre-trained Model for Knowledge Tracing). The PMKT model, as shown in Fig. 2,

features a design that integrates the Transformer and BERT architectures. It aims to effectively predict students' knowledge states through pre-training on large-scale student-answer interaction data. After completing the pre-training phase, all model parameters, except those in the output layer, are transferred to the fine-tuning stage for prediction tasks on small datasets. During this phase, the parameters of the output layer are reset after random initialization, while the parameters inherited from the pre-training phase are used to initialize the knowledge tracing tasks for small-scale target datasets. This process enables the model to further adjust to specific small sets of interaction data, adapting and integrating knowledge state representations from different systems.

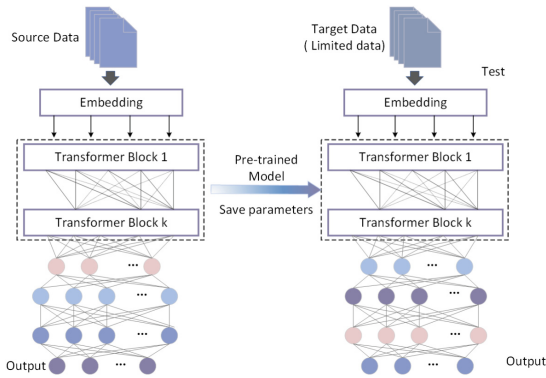


Fig. 2. Pre-trained model framework diagram

3.1 Transformer-Based Knowledge Tracing Model

The knowledge tracing model based on the Transformer architecture, aims to predict students' knowledge states by analyzing their complete historical performance (Fig. 3). It focuses on the structural characteristics of the questions and the time elapsed between solving steps. At the encoder, the student's interaction sequence $(0, x_1, x_2, \dots, x_{n-1})$ passes through the interaction mapping layer to generate a high-dimensional embedding $(0, e_1, e_2, \dots, e_{n-1})$. The sequence e_i and the interaction timestamp t_i go through Transformer blocks to learn contextual dependencies, in order to learn the hidden representation sequence h_i . At the decoder, the h_i sequence from the encoder, the embedded problem sequence e_{i+1} , and the timestamp t_{i+1} are processed together again through Transformer blocks to generate the decoder's hidden representation h_{i+1}^* .

The interactive mapping layer first creates an interaction-skill mapping matrix W and a skill embedding matrix S , and calculates the embeddings of the encoder-side input interaction x_i and the decoder-side input question q_i :

$$e_i = \text{softmax}(W_i \cdot) S \tag{1}$$

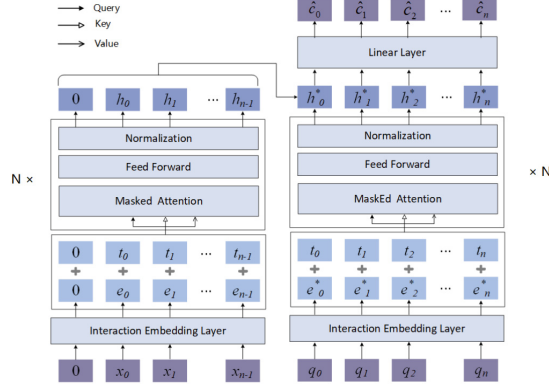


Fig. 3. Pre-trained model framework diagram

W_i represents the i -th row of matrix W , indicating the weights associated with all potential skills related to the interaction x_i . The *softmax* function normalizes these weights. Each column of matrix S represents a vector representation of a specific potential skill. Therefore, the static representation x_i of the interaction is the weighted sum of all potential skills.

In the attention masked layer, the contextualized interaction representation h_j is calculated from the static interaction representation e_j .

$$q_j = Q \cdot e_j, \quad k_j = K \cdot e_j, \quad v_j = V \cdot e_j \quad (2)$$

$$A_{ij} = \frac{q_j k_i + b(\Delta t_{j-i})}{\sqrt{d_k}} \quad (3)$$

$$h_j = \sum_{i \leq j} \text{softmax}(A_{ij}) v_i \quad (4)$$

The masked attention layer extracts the query q_j , key k_j , and value v_j by multiplying the static interaction representation e_j with three trainable matrices Q , K , and V , respectively. The key and query can be interpreted as latent skills associated with the interaction e_j , while the value represents the state of the latent skills related to e_j . To calculate the attention A_{ij} assigned to past interaction e_i , two components are considered: the degree of overlap $q_j k_i$ between the latent skills of interactions e_j and e_i , and the temporal interval deviation $b(\Delta t_{j-i})$.

3.2 BERT-Based Knowledge Tracing Model

BERT, the encoder module of the Transformer, integrates its architecture with deep knowledge tracing. The architecture, as illustrated, undergoes a training process where the correctness a_i in the interaction sequence (q_i, a_i) is randomly masked, meaning a_i is replaced with [MASK]. The model predicts the correctness of this small, randomly masked portion of the interaction based on the

bidirectional context within the sequence. The output module then provides the predicted probability of correctness for each interaction in the sequence, the model architecture is shown in Fig. 4. In the multi-head self-attention layer, each “head” is responsible for projecting the embedding of the input matrix X into the matrices Q , K and V . This is achieved through dot product operations with their respective trainable projection matrices, including W_Q , W_K , and W_V :

$$\begin{aligned}
 Q &= XW_Q \\
 K &= XW_K \\
 V &= XW_V
 \end{aligned}
 \tag{5}$$

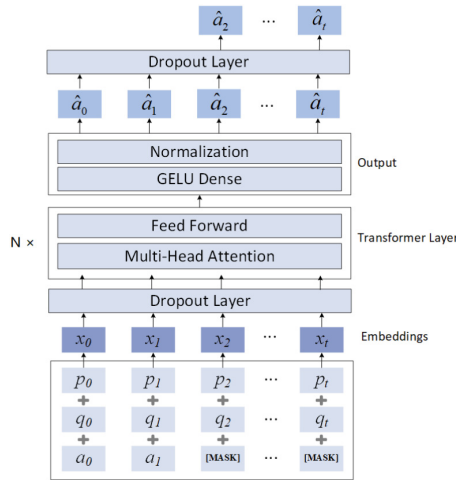


Fig. 4. Pre-trained model framework diagram

The intermediate dimension M for each head is $M' = \frac{M}{H}$. For the i -th self-attention head, with $1 \leq i \leq H$, the calculation can be expressed as follows:

$$A_{ij} = Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{M}}\right)V
 \tag{6}$$

The result $A_i \in R^{T \times M'}$. Then, all attention results across the H heads are concatenated in the output layer of the multi-head self-attention module:

$$Z = Concat(A_1, A_2, \dots, A_H)W_O
 \tag{7}$$

where $W_O \in R^{M \times M}$ is the weight matrix used to compute the final output embedding $Z \in R^{T \times M}$ from the multi-head attention module. The attention calculation for each head can be parallelized.

The output from the multi-head self-attention module is then fed into a position-wise fully connected feed-forward neural network:

$$FFN(Z) = \text{Max}(0, Z\Phi_1 + b_1)\Phi_2 + b_2 \quad (8)$$

where $\Phi_1 \in R^{M \times L}$ and $\Phi_2 \in R^{L \times M}$ are the trainable weight matrices for the hidden layer and output layer of the FNN module, respectively. Additionally, $b_1 \in R^L$ and $b_2 \in R^M$ are the bias vectors for these two layers. These trainable weight matrices and bias vectors are set specific to each layer of the transformer component.

To avoid potential cheating in the BERT prediction process, we divided the data into two parts during training: one part randomly masks 15% of the answers to enhance model performance, while the other part only masks the last answer in the sequence. During the testing phase, we only predict the correctness of the last masked answer.

4 Experiments

4.1 Datasets

We utilize various real datasets of ITS, summarized in Table 1.

ASSISTments datasets¹: ASSISTments 2009 (ASSIST09), ASSISTments 2012 (ASSIST12), and ASSISTments 2017 (ASSIST2017) datasets are sourced from the ASSISTments online tutoring platform for 2009, 2012, and 2017, serving as benchmarks for knowledge tracing (KT) methods, especially the ASSIST2009 dataset over the past decade.

EdNet²: This dataset is a stratified dataset, with each subset containing different types of student activities. It is one of the largest public datasets in the education field.

KDDcup Datasets³: It is prepared for the 2010 data mining competition, contain details such as exercise course hierarchy, KCs used, and student performance. They include Algebra 2005–2006 (Algebra05), Algebra 2006–2007 (Algebra06), and Bridge to Algebra 2006 (Bridge06).

Statics2011⁴: Statics2011 (Statics11) is collected from engineering mechanics courses in a college. The dataset comprises 335 students, 1,224 items, and 81 KCs, with an average of 568.45 responses per item. Following pre-processing, the dataset yielded 187,445 interactions, representing a reduction of 50% from the original dataset.

4.2 Experimental Setup

The experiments conducted in this study are categorized into eight groups, taking into consideration the size of the dataset used for both the pre-training and fine-tuning phases. To evaluate the performance of the pre-training approach, we ran a

¹ <https://sites.google.com/site/assistmentsdata/datasets>.

² <https://github.com/rriid/ednet>.

³ <http://pslclatashop.web.cmu.edu/KDDCup/downloads.jsp>.

⁴ <http://pslclatashop.web.cmu.edu/DatasetInfo?datasetId=507>.

Table 1. Statistics of datasets

| | Students | Questions | KCs |
|-----------|----------|-----------|-----|
| ASSIST09 | 3,841 | 15,911 | 123 |
| ASSIST12 | 27,405 | 47,104 | 265 |
| ASSIST17 | 1,709 | 4,117 | 102 |
| EdNet | 5,000 | 23,169 | 188 |
| Statics11 | 333 | 1,224 | 81 |
| Algebra05 | 575 | 173,113 | 112 |
| Algebra06 | 3,310 | 1,379 | 899 |
| Bridge06 | 1,146 | 129,263 | 493 |

5-fold student stratified cross-validation on various datasets. The model’s performance was evaluated using the Area Under the Curve (AUC), which comprehensively considers the model’s accuracy and recall, balancing predictive capability and efficiency. Specific experimental conditions are detailed in Table 2.

The subscript indicates the number of original datasets combined in the experiment, while the superscript indicates the required platform on which the datasets were collected; * means any platform is deemed acceptable. For example, D_2^* means two datasets are selected for this experiment, and each dataset is acceptable. On the other hand, D_1^{assist} indicates that only one dataset can be selected, and it must be one of these datasets collected on ASSISTments, i.e., ASSISTments2009, ASSISTments2012, or ASSISTments2017. E is a distinct dataset that is not included in D . Especially, the subscript $lite(n)$ indicates that the dataset is a subset consisting of n (which may vary in different experiments) interactions. Moreover, “SOTA-Model” refers to the state-of-the-art KT models, which may be compared with the pre-trained KT models. “Nonfreezing” indicates that the parameters are not frozen during the fine-tuning. It implies that all the parameters from the pre-trained model used for training are retained.

Table 2. The experimental settings

| | Pre-Training Data | Target data | Contrast Experiment |
|---|-----------------------------|----------------------------|-----------------------|
| 1 | D_1^* | $E_{lite(5,000)}^*$ | Baseline |
| 2 | D_2^* | $E_{lite(5,000)}^*$ | Baseline, SOTA-Model |
| 3 | D_{all} | $E_{lite(5,000)}^*$ | Baseline |
| 4 | D_2^* | E_1^* | Baseline, SOTA-Model |
| 5 | D_1^{assist} | $E_{lite(5,000)}^{assist}$ | Baseline |
| | $D_1^* - D_{lite(5,000)}^*$ | $D_{lite(5,000)}^*$ | Baseline |
| 6 | D_1^* | $E_{lite(n)}^*$ | Baseline |
| 7 | D_1^* | $E_{lite(50 100)}^*$ | Baseline, SOTA-Model |
| 8 | D_2^*, D_{all} | $E_{lite(5,000)}^*$ | Baseline, Nonfreezing |

In the subsequent empirical findings, “baseline” refers to the performance of the training on the respective dataset without the utilization of pre-training. The bold numbers are the best performance.

4.3 Experiment Results and Analysis

Experiment 1. A comparative experiment was carried out by randomly selecting two datasets, one for pre-training and the other for fine-tuning with 5000 interactions. As shown in Table 3, using a pre-trained model significantly improved predictive performance compared to training only on a small-scale dataset. The largest performance increase was 3.83%.

Table 3. Cross-Training Experiment Results with Homogeneous Data (AUC)

| Model | Pre-Training | Fine-tuning | | |
|-------------|--------------|---------------|---------------|---------------|
| | | Algebra06 | ASSIST09 | EdNet |
| BERT | Baseline | 0.6261 | 0.5969 | 0.6013 |
| | Algebra06 | – | 0.6263 | 0.6280 |
| | ASSIST09 | 0.6372 | – | 0.5841 |
| | EdNet | 0.6439 | 0.6103 | – |
| Transformer | Baseline | 0.8024 | 0.8416 | 0.8677 |
| | Algebra06 | – | 0.8520 | 0.8691 |
| | ASSIST09 | 0.8393 | – | 0.8638 |
| | EdNet | 0.8132 | 0.8283 | – |

Experiment 2. For pre-training, two datasets from ASSIST09, Algebra06, and EdNet were chosen, with a small set of 5000 interactions from another dataset used for fine-tuning. Results in Table 4, show our pre-training method boosts knowledge tracing models’ performance on small datasets. Pre-training with the Transformer on EdNet and ASSIST09, then fine-tuning on 5000 Algebra06 interactions, increased AUC by 5.07% over the baseline. Further experiments confirmed pre-training’s effectiveness on small datasets, as shown in Table 5.

Table 4. The results on two datasets (AUC)

| Model | Pre-Training | Fine-tuning | | |
|-------------|---------------------|---------------|---------------|---------------|
| | | Algebra06 | ASSIST09 | EdNet |
| BERT | Baseline | 0.6261 | 0.5969 | 0.6013 |
| | EdNet, ASSIST09 | 0.6319 | – | – |
| | EdNet, Algebra06 | – | 0.6137 | – |
| | Algebra06, ASSIST09 | – | – | 0.6265 |
| Transformer | Baseline | 0.8024 | 0.8416 | 0.8677 |
| | EdNet, ASSIST09 | 0.8531 | – | – |
| | EdNet, Algebra06 | – | 0.8675 | – |
| | Algebra06, ASSIST09 | – | – | 0.8850 |

Table 5. The results from baselines (AUC)

| Models | Algebra06 |
|-------------|-----------|
| BKT | 0.5915 |
| BKT+Forgets | 0.5922 |
| SAKT | 0.6582 |

Experiment 3. The experiment combines datasets in pairs for pre-training using the BERT model. The paper selected 5000 interactions from the Static2011 and ASSIST09 datasets for fine-tuning to test our method. The results, shown in Table 6, reveal that the model pre-trained on the combined datasets significantly surpasses the non-pre-trained model in AUC. The AUC on the ASSIST09 dataset increased by 3.05%, and on the Static2011 dataset by 3.32%.

Table 6. The results of large-datasets (AUC)

| Pre-Training | Fine-tuning | Baseline | Ours |
|--|-------------|----------|---------------|
| ASSIST12, ASSIST17, Algebra05, Algebra06, EdNet | ASSIST09 | 0.5969 | 0.6274 |
| ASSIST09, ASSIST12, ASSIST17, EdNet, Spanish, Algebra05, Algebra06 | Statics11 | 0.7648 | 0.7980 |

Experiment 4. Based on the conclusion of Experiment 2, the paper utilized pre-training methods to verify the results on the complete dataset. Specifically, we randomly selected two datasets from ASSIST09, Algebra06, and EdNet for pre-training, fine-tuned on another complete dataset, and compared our method with other knowledge tracing models. The results are presented in Table 7.

Table 7. The results on the complete dataset (AUC)

| Models | Algebra06 | ASSIST09 | EdNet |
|-----------------------------|---------------|---------------|---------------|
| BKT | 0.6707 | 0.6975 | 0.6692 |
| BKT+Forgets | 0.6844 | 0.7306 | 0.6894 |
| DKT | 0.7925 | 0.7570 | 0.7791 |
| SAKT | 0.7637 | 0.7991 | 0.7683 |
| Bert-based | 0.7902 | 0.7544 | 0.7521 |
| Pre-KT (Bert) | 0.8093 | 0.7862 | 0.7525 |
| Transformer-based | 0.8036 | 0.8002 | 0.7690 |
| Pre-KT (Transformer) | 0.8144 | 0.8277 | 0.7815 |

Experiment 5. To further investigate the effectiveness of our pre-training technology, a series of experiments were performed, training models on various datasets from the same ITS, as shown in Table 8. One of the ASSISTments datasets was selected as the pre-training data, and 5000 interactions on the other two datasets. Furthermore, additional experiments were conducted on the same dataset, in which a random subset of 5000 interactions was used for fine-tuning, and the remaining interactions were used for pre-training, as illustrated in Table 9.

Table 8. The results from the similar datasets (AUC)

| Model | Pre-Training | Fine-tuning | | |
|-------------|--------------|---------------|---------------|---------------|
| | | ASSIST09 | ASSIST12 | ASSIST17 |
| BERT | Baseline | 0.5969 | 0.7609 | 0.6670 |
| | ASSIST09 | – | 0.7629 | 0.6593 |
| | ASSIST12 | 0.5662 | – | 0.6713 |
| | ASSIST17 | 0.5778 | 0.7652 | – |
| Transformer | Baseline | 0.8416 | 0.7785 | 0.7950 |
| | ASSIST09 | – | 0.7829 | 0.7931 |
| | ASSIST12 | 0.8417 | – | 0.7994 |
| | ASSIST17 | 0.7936 | 0.7815 | – |

Table 9. The results from the same datasets (AUC)

| Model | Datasets | Pre-Training Numbers | Fine-Tuning Numbers | Baseline | Ours |
|-------------|-----------|----------------------|---------------------|----------|---------------|
| BERT | ASSIST09 | 277,072 | 5000 | 0.5645 | 0.5296 |
| | Algebra06 | 1803,534 | 5000 | 0.6347 | 0.6411 |
| Transformer | ASSIST09 | 277,072 | 5000 | 0.7404 | 0.7349 |
| | Algebra05 | 602,015 | 5000 | 0.7276 | 0.7323 |

Experiment 6. This experiment aims to investigate the impact of dataset size on performance. Firstly, the BERT model was pretrained on the ASSIST09 dataset and subsequently fine-tuned on the EdNet dataset, which contains interaction sequences of different sizes. Similarly, the Transformer model was pretrained on the EdNet dataset and further fine-tuned on the ASSIST09 dataset. Figure 5 shows that the number of interaction terms used for fine-tuning will affect the results. Figure 6 displays the difference in AUC between pre-trained and non-pre-trained models, more intuitively demonstrating that the impact of pre-training is more pronounced when approximately 1/3 of the EdNet and ASSIST09 datasets are selected.

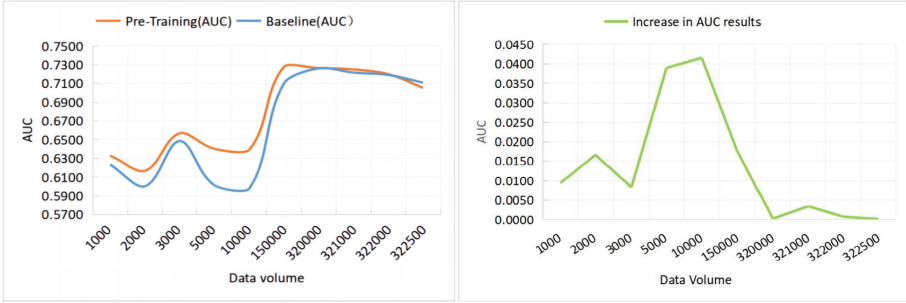


Fig. 5. Performance of different data amounts (BERT-based)

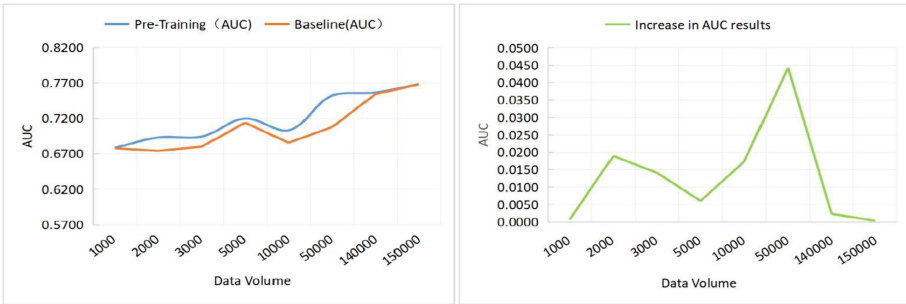


Fig. 6. Performance of different data amounts (Transformer-based)

Experiment 7. In this experiment, we pre-trained the model on the EdNet dataset and then fine-tuned it on the ASSIST09 dataset. As shown in Table 10, the results indicate that our proposed pre-training method also performs well on 100 interactions of the ASSIST09 dataset. Moreover, pre-training on the Algebra05 dataset improved the AUC by 5.15% on 50 interactions of the ASSIST09 dataset compared to training on the small dataset alone, which can be considered a good solution to the “limited data” challenge initially identified. Further experiments were conducted on classical knowledge tracing models using the same 100 interactions of the ASSIST09 dataset, as shown in Table 11.

Table 10. The results on the smaller datasets (AUC)

| Pre-Training | Fine-tuning | | |
|--------------|----------------|----------------|---------------|
| | ASSIST09 (200) | ASSIST09 (100) | ASSIST09 (50) |
| Baseline | 0.5380 | 0.6064 | 0.6107 |
| EdNet | 0.6117 | 0.6395 | 0.6104 |
| Baseline | 0.5254 | 0.6185 | 0.6152 |
| Algebra05 | 0.5549 | 0.6492 | 0.6667 |

Table 11. The results of the 100 pieces of data on the baselines

| Models | AUC |
|-------------|--------|
| BKT | 0.5364 |
| BKT+Forgets | 0.5686 |
| DKT | 0.4736 |
| SAKT | 0.3967 |

Experiment 8. Parameter freezing is a valuable technique in the fine-tuning of intricate models as it effectively reduces the number of parameters requiring adjustment during the training. Two sets of experiments (Experiment 2 and Experiment 3) demonstrated significant improvements in AUC compared to Experiments 1–7. During the fine-tuning phase, a decision was made to freeze a portion of the training parameters, specifically limiting the training to only two weight parameters, namely `emb_pid` (question) and `emb_r` (correct). The performance of these experiments is presented in Table 12. However, based on our experimental settings, it was observed that parameters freezing did not yield any improvement in the experimental results.

Table 12. The results of Freezing parameter (AUC)

| Model | Settings | Baseline | Nonfreezing parameter | Freezing parameter |
|-------------|--|----------|-----------------------|--------------------|
| BERT | Pre-Training: ASSIST09, ASSIST12, ASSIST17, EdNet, Spanish, Algebra05, Algebra06 Fine-Turning: Statics11 | 0.7648 | 0.7980 | 0.7602 |
| Transformer | Pre-Training: EdNet, ASSIST09 Fine-Turning: Algebra06 | 0.8024 | 0.8531 | 0.8130 |

5 Conclusions and Future Work

The main topic of this article is predicting future learner responses based on limited amounts of interaction. The knowledge tracing models are pre-trained on one or multiple comprehensive datasets that are publicly accessible. The pre-training models BERT and Transformer have been employed to capture features pre-training to the knowledge states of students based on limited data. The features obtained from the pre-training model on these datasets were utilized

by reusing the weight parameters. Various experiments have been conducted to arrive at the following conclusions:

- In cases where the quantity of data is restricted, pre-training techniques have been found to be more efficacious for knowledge tracing. The AUC for all parameters of the pre-trained model is greater than that of the fixed part parameters.
- The Transformer model-based pre-training is a more appropriate approach for predicting future learner responses with limited data, as opposed to the BERT model that employs a random masking technique on the sequence.
- The source of the pre-training data and fine-tuning data, both of which originate from the same ITS, have little effect on the experimental results predicting future learner responses.
- The number of interactions used for fine-tuning has an impact on the results, which will be more apparent when approximately 1/3 of the data from the EdNet and ASSIST09 datasets is selected. However, further research is needed in the future to investigate the phenomenon of a decrease in AUC value after increasing the amount of fine-tuning data.
- The incorporation of pre-training before making predictions leads to a significant improvement in the AUC for the entire dataset. In comparison to alternative knowledge tracing models, the knowledge tracing model that incorporates pre-training demonstrates superior predictive performance.

Drawing from the aforementioned findings, our future research endeavors will focus on addressing the “cold start” predicament within the realm of knowledge tracing. To this end, we intend to enhance the predictive efficacy on modest datasets by employing diverse pre-training techniques and fine-tuning approaches to extract salient features such as questions and KCs. Furthermore, the paper aspires to attain elevated prediction accuracies through pre-training, even in scenarios where the data is severely restricted, for instance, with only 10 or a mere 1 data point.

Acknowledgments. This work was supported by the Science and Technology Project of Gansu (21YF5GA102, 21YF5GA006, 21ZD8RA008, 22ZD6GA029, 22YF7GA003), Gansu Key Talent Project (11256471037), the Fundamental Research Funds for the Central Universities (lzujbky-2022-ct06), Supercomputing Center of Lanzhou University.

References

1. Abdelrahman, G., Wang, Q.: Learning data teaching strategies via knowledge tracing. *Knowl.-Based Syst.* **269**, 110511 (2023)
2. Anderson, J.R., Boyle, C.F., Reiser, B.J.: Intelligent tutoring systems. *Science* **228**(4698), 456–462 (1985)
3. Baker, R.S.J., Corbett, A.T., Aleven, V.: More accurate student modeling through contextual estimation of slip and guess probabilities in Bayesian knowledge tracing. In: Woolf, B.P., Aïmeur, E., Nkambou, R., Lajoie, S. (eds.) *ITS 2008*. LNCS,

- vol. 5091, pp. 406–415. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69132-7_44
4. Choi, Y., et al.: Towards an appropriate query, key, and value computation for knowledge tracing. In: Proceedings of the Seventh ACM Conference on Learning@Scale, pp. 341–344 (2020)
 5. Corbett, A.T., Anderson, J.R.: Knowledge tracing: modeling the acquisition of procedural knowledge. *User Model. User-Adap. Inter.* **4**, 253–278 (1994)
 6. Ghosh, A., Heffernan, N., Lan, A.S.: Context-aware attentive knowledge tracing. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2330–2339 (2020)
 7. Liu, Z., Chen, J., Luo, W.: Recent advances on deep learning based knowledge tracing. In: Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, pp. 1295–1296 (2023)
 8. Nakagawa, H., Iwasawa, Y., Matsuo, Y.: Graph-based knowledge tracing: modeling student proficiency using graph neural network. In: IEEE/WIC/ACM International Conference on Web Intelligence, pp. 156–163 (2019)
 9. Pandey, S., Karypis, G.: A self-attentive model for knowledge tracing. arXiv preprint [arXiv:1907.06837](https://arxiv.org/abs/1907.06837) (2019)
 10. Pandey, S., Srivastava, J.: RKT: relation-aware self-attention for knowledge tracing. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp. 1205–1214 (2020)
 11. Piech, C., et al.: Deep knowledge tracing. In: Advances in Neural Information Processing Systems, vol. 28 (2015)
 12. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al.: Improving language understanding by generative pre-training (2018)
 13. Sha, L., Hong, P.: Neural knowledge tracing. In: Frasson, C., Kostopoulos, G. (eds.) BFAL 2017. LNCS, vol. 10512, pp. 108–117. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67615-9_10
 14. Shin, D., Shim, Y., Yu, H., Lee, S., Kim, B., Choi, Y.: SAINT+: integrating temporal features for EdNet correctness prediction. In: LAK21: 11th International Learning Analytics and Knowledge Conference, pp. 490–496 (2021)
 15. Song, K., Tan, X., Qin, T., Lu, J., Liu, T.Y.: MASS: masked sequence to sequence pre-training for language generation. arXiv preprint [arXiv:1905.02450](https://arxiv.org/abs/1905.02450) (2019)
 16. Song, X., Li, J., Lei, Q., Zhao, W., Chen, Y., Mian, A.: Bi-CLKT: Bi-graph contrastive learning based knowledge tracing. *Knowl.-Based Syst.* **241**, 108274 (2022)
 17. Sonkar, S., Waters, A.E., Lan, A.S., Grimaldi, P.J., Baraniuk, R.G.: qDKT: question-centric deep knowledge tracing. arXiv preprint [arXiv:2005.12442](https://arxiv.org/abs/2005.12442) (2020)
 18. Tato, A., Nkambou, R.: Deep knowledge tracing on skills with small datasets. In: Crossley, S., Popescu, E. (eds.) ITS 2022. LNCS, vol. 13284, pp. 123–135. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-09680-8_12
 19. Tong, S., et al.: Structure-based knowledge tracing: an influence propagation view. In: 2020 IEEE International Conference on Data Mining (ICDM), pp. 541–550. IEEE (2020)
 20. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
 21. Villano, M.: Probabilistic student models: Bayesian belief networks and knowledge space theory. In: Frasson, C., Gauthier, G., McCalla, G.I. (eds.) ITS 1992. LNCS, vol. 608, pp. 491–498. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-55606-0_58
 22. Weston, J., Chopra, S., Bordes, A.: Memory networks. arXiv preprint [arXiv:1410.3916](https://arxiv.org/abs/1410.3916) (2014)

23. Wulf, J., Blohm, I., Leimeister, J.M., Brenner, W.: Massive open online courses. *Bus. Inf. Syst. Eng.* **6**, 111–114 (2014)
24. Yang, H., Cheung, L.P.: Implicit heterogeneous features embedding in deep knowledge tracing. *Cogn. Comput.* **10**, 3–14 (2018)
25. Yang, Y., et al.: GIKT: a graph-based interaction model for knowledge tracing. In: Hutter, F., Kersting, K., Lijffijt, J., Valera, I. (eds.) *ECML PKDD 2020*. LNCS, vol. 12457, pp. 299–315. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-67658-2_18
26. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R.R., Le, Q.V.: XLNet: generalized autoregressive pretraining for language understanding. In: *Advances in Neural Information Processing Systems*, vol. 32 (2019)
27. Ye, Y., Shan, Z.: HGKT: hypergraph-based knowledge tracing for learner performance prediction. In: *2023 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9. IEEE (2023)
28. Zhang, J., Shi, X., King, I., Yeung, D.Y.: Dynamic key-value memory networks for knowledge tracing. In: *Proceedings of the 26th International Conference on World Wide Web*, pp. 765–774 (2017)
29. Zhang, L., Xiong, X., Zhao, S., Botelho, A., Heffernan, N.T.: Incorporating rich features into deep knowledge tracing. In: *Proceedings of the Fourth ACM Conference on Learning@ Scale*, pp. 169–172 (2017)
30. Zhao, J., Bhatt, S., Thille, C., Gattani, N., Zimmaro, D.: Cold start knowledge tracing with attentive neural Turing machine. In: *Proceedings of the Seventh ACM Conference on Learning@ Scale*, pp. 333–336 (2020)



Chorus: More Efficient Machine Learning on Serverless Platform

Guang Yang^{1,2}, Jie Liu^{1,2,3,4}(✉), Shuai Wang^{1,2}, Muzi Qu^{1,2}, Dan Ye^{1,2},
and Hua Zhong^{1,2,3}

¹ Institute of Software, Chinese Academy of Science, Beijing 100190, China
{yangguang17, ljie, wangshuai, qumuzi18, yedan, zhonghua}@otcaix.iscas.ac.cn

² University of Chinese Academy of Sciences, Beijing 100049, China

³ Key Laboratory of System Software (Chinese Academy of Sciences) and State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China

⁴ University of Chinese Academy of Sciences, Nanjing, Nanjing 211135, China

Abstract. Executing machine learning jobs on serverless platform could gain higher performance in a simplified manner. But current solutions have not fully utilized the flexibility of serverless computing. The widely used Bulk Synchronous Parallel (BSP) mode has significant resource waste, the parameter server nodes suffer bottleneck pressure from both networking and performance aspects. This paper presents Chorus, a machine learning framework on serverless platform based on the parameter server architecture. In Chorus we propose the Lambda Synchronous Parallel (LSP) model to coordinate Lambda workers. It simultaneously utilizes functions with different resource level to collaborate and dynamically adjusts resource allocation to maintain balance of model training. To alleviate the bottleneck pressure on the parameter server, we build a buffer system in memory database to exchange gradient data between the parameter server and workers. We set up multiple buffer slots in the buffer system to alleviate network pressure and design a buffer merging strategy to disperse computational pressure of parameter server among Lambda workers. In the experiments on different ML algorithms with different synchronous parallel models, Chorus shows outstanding performance improvements and budget-saving capacities.

Keywords: Serverless · Machine Learning · Distributed Computing

1 Introduction

Machine learning is a foundational technology in data processing that has found extensive application across various scientific and industrial domains, such as text analysis, search engines, document classification, and recommendation systems. Its widespread adoption has positioned machine learning as a core computing task in modern data centers. With the rapid advancements in distributed computing, an increasing number of machine learning jobs can now be executed in cloud environments. However, managing distributed

environments remains significantly more complex for non-computer science experts compared to managing a single machine. Running tasks in the cloud involves challenges such as resource allocation and worker coordination, leading to resource shortages or wastage in many cases. These challenges impede the widespread adoption of running machine learning jobs using distributed computing.

Recently, serverless architecture has gained significant attention as a new paradigm in cloud services. Within a serverless architecture, users directly execute their functions without managing the underlying cluster resources. The system handles resource provisioning and scales computing power transparently, allowing users to focus solely on their program logic. Researchers have migrated many types of tasks to serverless platform, including data analysis [1, 2], linear algebra processing [3], video encoding [4, 5], and code compiling [6], etc. Serverless architecture also presents an opportunity to simplify and economize the processing of machine learning jobs. In this domain, the parameter server architecture [7] is the most appropriate choice in serverless environment. In this architecture, worker nodes operate independently without communicating to each other in most cases. However, to ensure the correctness and efficiency of the training process, proper coordination of worker nodes is essential, typically achieved through the synchronous parallel model. There are three common synchronous parallel models: the Bulk Synchronous Parallel (BSP) model, Asynchronous Synchronous Parallel (ASP) model, and Stale Synchronous Parallel (SSP) model [8], each offering distinct advantages and trade-offs in managing distributed machine learning tasks.

Many researchers have already proposed frameworks for training machine learning models on serverless platform [9–15]. However, we have identified several significant issues with the existing solutions. 1) Almost all frameworks are using the BSP model, which converges stably but lacks performance. 2) The parameter server nodes need to exchange data with all workers and handle all model update tasks, making them very prone to become bottlenecks of the entire system. MLLess [13] tries to solve this issue by using Lambda functions as parameter server, but introduces redundant computation. Lang Feng [14] solve it by gradients merging, but only supports BSP model.

To perform machine learning in serverless platform and overcome these weaknesses, we present Chorus, a machine learning framework in serverless computing. Chorus implements the parameter server architecture and utilizes advantages of serverless computing to make the execution more efficient and economical.

In the design of Chorus, we leverage the resource scheduling flexibility of serverless computing to enable functions with different resource level to collaborate and dynamically adjust resource allocation. This approach ensures a balanced and efficient training process, thus saving time and budget for the uses. We build a memory buffering system for gradient data that alleviates network pressure through a multi-buffer slot approach and disperses the computational pressure on the parameter server through a buffer merging strategy. We designed separate strategies for buffer reading priority and buffer merging priority to ensure that the buffering strategy does not disrupt the balance of model training. By alleviating the bottleneck issue on the parameter server, the overall system performance has significantly improved.

We implement Chorus in AWS and evaluate it with different ML algorithms and synchronous parallel models. Chorus shows significant performance improvement and a favorable effect in budget saving.

In summary, the main contributions we make in this research are the followings:

- We propose a novel synchronous parallel model specialized for ML jobs under serverless architecture. It accelerates model convergence without adding budget.
- We propose a memory buffer system for exchanging gradient data, and design a buffer merging strategy along with corresponding priority algorithms. This approach effectively addresses the bottleneck issue on parameter server while minimizing its impact on training effectiveness.
- We implement our framework and evaluate it with different ML algorithms and synchronous parallel models, proving our framework has a better computing performance and budget-saving capacity.

2 Background and Motivation

In this section, we give a brief introduction of the background knowledge of our work and explain the motivations of the innovations in Chorus.

Most ML training tasks can be abstracted to an optimization problem, that is solving the extreme value of a target function. Gradient descent is one of the most popular methods, it iteratively adjusts the parameter set, moving in the direction of the steepest gradient until it reaches or gets close enough to an extreme point. The gradient descent algorithm is easy to implement in distributed environment, a typical implementation is the parameter server architecture with data parallel model. In this architecture, the training set is split into shards, and workers compute the gradient of every shard and send the result to one or several central nodes called parameter server. The parameter server receives all gradients and updates the model. Therefore, the parameter server nodes bear both computing and network traffic pressure, they are very prone to be the bottleneck of the system. Alleviating this bottleneck issue is the first motivation of our work.

To ensure model convergence and improve efficiency, each worker must be coordinated properly. If the algorithm is implemented with complete equivalence, the model can only be updated after all workers have completed one iteration, this is the Bulk Synchronous Parallel (BSP) model. Figure 1(a) shows the process of BSP model. In BSP model, all workers run synchronously, the parameter server updates model only after it received the gradients from every worker, the period is called one epoch. BSP model has the best and most rigorous convergence effect, but also has obvious resource waste. To improve the performance of the training system, more parallel models are proposed.

The Asynchronous Synchronous Parallel (ASP) model removes all the waiting procedures in the training process and gives up the guarantee of model convergence. The parameter server updates the model instantly after they receive one gradient from any worker, and the worker starts its next iteration immediately after it gets a new parameter set from parameter server. There is no epoch boundary in ASP model. Figure 1(b) shows the synchronous process of workers in ASP model.

The ASP model improves the system resource utilization ratio significantly, but it also affects the convergence of the algorithm. Sometimes it even makes the ML model

unable to converge. The Stale Synchronous Parallel (SSP) model [8] is a compromise between BSP and ASP model. The SSP mode introduces the concept of *staleness*, which represents the iteration counts distance between the fastest and the slowest worker. The SSP mode allows users to set a staleness threshold, ensuring that the staleness during the training process is always kept below the threshold. In other words, SSP model blocks the fast workers which reach the threshold and waits for the stragglers. Figure 2 shows the synchronous process of workers in SSP model.

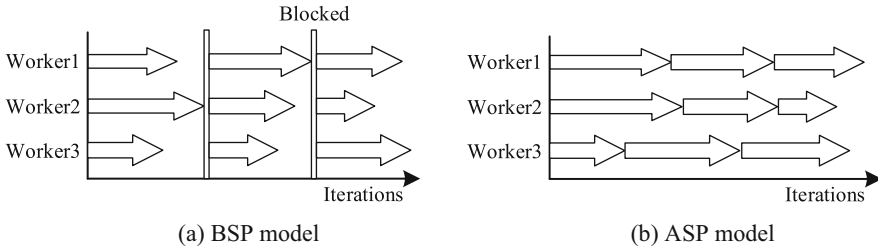


Fig. 1. Bulk Synchronous Parallel (BSP) model.

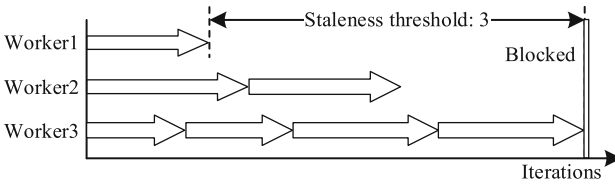


Fig. 2. Stale Synchronous Parallel (SSP) model.

It can be deduced that when the staleness threshold is set to 0, the SSP model degrades to BSP model; and when the it tends toward infinity, the SSP model tends toward ASP model. The staleness level affects model convergence significantly, especially under the circumstance when the training is hard to converge. To illustrate the influence of staleness in the training process, we implement a testing parameter server architecture with BSP, ASP, and SSP model. We use the logistic regression algorithm and choose a relatively high learning rate to make the convergence harder, then perform training with different staleness thresholds. The testing results are shown in Fig. 3.

In the graph, BSP converges the fastest while ASP converges the slowest, and among SSP modes, the lower of staleness threshold, the faster the model converges over iteration (SSP-1 almost overlaps with BSP). However, lower threshold may increase the training time because it can lead to more blocking. Therefore, we need to find a balance between reducing blocking and accelerating convergence to improve training speed and save budget. This is the second motivation of our work.

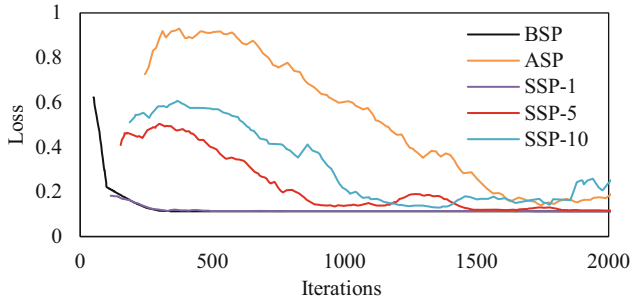


Fig. 3. Comparisons of SSP model with different staleness threshold, where “SSP-n” indicates SSP model with staleness threshold n, the x-axis is the sum of iteration numbers of each shard.

3 Lambda Synchronous Parallel (LSP) Model

The SSP model accelerates model convergence compared to ASP, but still remains some blocking time. We make further improvements to SSP model, which is the Lambda Synchronous Parallel (LSP) model. The LSP model is based on the SSP model and inherits its staleness rules. Compared to SSP, during task execution, LSP dynamically replaces the running instances of functions, thus adjusting the resource allocation of functions. The purpose of instance replacing is to skew resource allocation towards lagging workers. We will illustrate the mechanism of LSP through an example shown in Fig. 4.

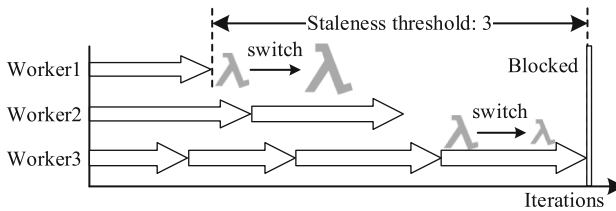


Fig. 4. Lambda Synchronous Parallel (LSP) model.

In this example, the staleness threshold is set to 3. Worker 3 is the most advanced in progress in the graph, with a 3-iteration distance compared to the most lagging Worker 1, reaching the staleness threshold. As a result, Worker3 will be blocked to wait for Worker1, triggering a function instance replacement action, which includes two simultaneous actions:

1. Close the function instance of Worker3, start a new instance with fewer allocated resources, read training data shard, and wait for the blocking to end and then continue with Worker3’s iteration task.
2. Start a new instance with more allocated resources than Worker1, read training data shard of Worker1, wait for Worker1 to finish the current iteration. When it finishes, close the instance of Worker1, and continue Worker1’s iteration with the new instance.

In Fig. 4, the bigger “ λ ”s indicate instances with more resources, and the smaller ones indicate instances with less resources. Theoretically, instances with more resources shall executes faster. Therefore, in the subsequent iterations, Worker3 will gradually slow down its unnecessary running speed and saving costs. These saved costs will be allocated to Worker1, allowing it to catch up, thus reducing the iteration progress difference between functions, and reducing blocking time in training process. LSP only performs instance replacement on blocked functions, and each time it replaces the fastest and slowest function’s instances simultaneously.

Instance replacement actions come at a cost. Although serverless functions are lightweight, starting a new function and loading training data still costly. LSP only replaces instances for the fastest (blocked) and slowest (lagging) functions, and the instance replacement actions always start when blocking occurs, utilizing the blocking time for replacement and preparing new instances in advance for lagging functions, minimizing the impact on the performance of whole system. From the actual execution flow of tasks, LSP is essentially the same as SSP, it only changes the time taken for each step, so it does not affect the correctness of computations and provides the same convergence guarantee as SSP.

The LSP mode can be formalized. We first define the *shard version* as an integer v . For the i th shard, its version v_i represents the iteration times of it. Then, we can define the *model or gradient version* as a vector:

$$\text{version} = (v_1, v_2, \dots, v_n) \quad (1)$$

where n is the number of shards, and v_i represents how many updates from the i th shard are included in this model or gradient. Obviously, for a training job contains m iterations, its model version will start from $(0, 0, \dots, 0)$ and keep increasing until reach (m, m, \dots, m) . When all shard versions reach m , the job completes.

The purpose of instance replacement is to narrow the iteration gap between the fastest worker and the slowest worker, so we set the resources allocated to a new worker proportional to the distance between its iteration count and the average version of all shards. Formally, for a worker in the j th iteration, assuming the model version at this time is (v_1, v_2, \dots, v_n) , the resource quantity allocated to the new worker in the next iteration can be expressed as:

$$R = r + \alpha \left(\frac{1}{n} \sum_{i=1}^n v_j - j \right) \quad (2)$$

where R indicates the resource quantity of the next instance, r is the base quantity of resources and should be constant, proportional constant α is an empirical value. In real practice, resource R can be the memory size or CPU cores of serverless function. To simplify the function deployment, the resource quantity should be discretized, and can be expressed as $R' = \lfloor cR/c \rfloor$. Where c represents the degree of discreteness.

During the scheduling process, instance replacements always occur in pairs. Equation (2) ensures that the reduction in resource quantity for the fastest instance is equal to the increase in resource quantity for the slowest instance, thereby the fee rate of the whole system is kept stable.

4 Buffering in Parameter Server Model

The bottleneck issue in the parameter server model hinders its application in serverless computing. This issue manifests in two aspects: network bottlenecks and performance bottlenecks with the parameter server. We propose a buffer system to solve these two aspects at the same time.

We utilize a in memory database service to store the ML model, and build a multi-slot buffer space in it for the parameter server and workers to exchange gradients. The number of slots is user-specified and fixed. If one slot is occupied, the worker can choose another empty one. The throughput of memory database is much bigger than few parameter nodes, it can accelerate the speed at which workers write gradient data, but it cannot accelerate the speed at which the parameter server consumes gradient data. Under the circumstance of massive concurrency, the buffer slot spaces will eventually be exhausted, and the queuing of workers on the parameter server will still occur constantly. To solve this issue, we design a buffer (gradient) merging strategy. We allow the gradients in buffer space to be merged before being fetched by the parameter server. For a worker which finished one iteration, if all buffer slots are occupied, the worker chooses a buffer slot via a priority algorithm and merges its gradient data into the buffer, then read the model parameter and proceed to its next iteration. However, buffer merging will interfere with the order of model updates. Without proper control, the buffer merging strategy could exacerbate the straggler problem seriously, because some gradient data may be “forgotten” in the buffer for a long time, and encumber the convergence of the model. To reduce this impact, we also design a buffer reading priority strategy for the parameter server to choose the proper buffer to read and avoid training imbalance. We will detail these two strategies below.

Buffer Reading Priority. The principle of this strategy is to always choose the “most urgently needed” data to update the model, and the “most urgently” is judged by the shard which have the smallest version. This rule above can be described formally. Assuming the model version is $V = (v_1, v_2, \dots)$, and a gradient has version $U = (u_1, u_2, \dots)$. First, select all the non-zero components in U , then find the corresponding components in model version V of each shard, form a new set V' :

$$V' = \{v_x | v_x \in V, u_x \neq 0\} \quad (3)$$

Calculate the minimum shard version in the set, record as $\min V = \min(V')$.

Then, for a group of gradients $U_1, U_2, U_3 \dots$, calculate each $\min V$ of them, record as $\min V_1, \min V_2, \min V_3, \dots$, the index of selected buffer can be expressed as:

$$\underset{i}{\operatorname{argmin}}(\min V_1, \min V_2, \min V_3, \dots) \quad (4)$$

This strategy ensures that the most lagging gradient data in the buffer space will be read first, thus avoids the increasement of staleness gap.

Buffer Merging Priority. First, we formally defined the version of the merged gradient as the sum of versions of two original gradients. For instance, if we merge a gradient of version (0, 0, 1, 0) with another gradient of version (1, 0, 1, 0), we get a new gradient of

version (1, 0, 2, 0). When a worker just finishes one iteration and found no vacant buffer slot left, it calculates the $\min V$ value for its new gradient and all other gradients in the buffer slots. The gradient in the buffer whose $\min V$ has the smallest value distance to the new gradient will be chosen for merging. Formally, assuming gradients $U_1, U_2, U_3 \dots$ are stored in buffer, and a new gradient U_0 is finding a merging target. Then the index of the chosen gradient can be expressed as:

$$\operatorname{argmin}_i (|\min V_i - \min V_0|) \quad (5)$$

where $i = 1, 2, 3 \dots$, and $\min V_0$ indicates the $\min V$ of U_0 .

This strategy will make the gradients prone to cluster in buffer slots by $\min V$, which is the most urgently needed data by the model. Gradients have similar $\min V$ values are prone to gather in one slot. When the parameter server reads the slot, it will get a gradient with relatively similar shard version, thus avoid training imbalance to some extent. The strategy also disperses the buffer merging operations across slots, avoiding conflict or queuing, thus unleashing the power of distributed computing as much as possible. The buffer merging strategy reduces the frequency of model updates, relieving the pressure on the parameter server and alleviating its computational bottleneck significantly.

5 Design of Chorus

With the LSP model and buffering system, we build Chorus, a framework for machine learning on serverless platform. We describe the design of Chorus in this section.

5.1 Architecture Overview

Chorus is built on AWS cloud, aside from the parameter server VM and serverless functions, Chorus utilizes two extra public services in the cloud. Figure 5 shows the system architecture of Chorus.

The system consists of four parts:

- Parameter server on EC2, a virtual machine to maintain the ML model and coordinate the serverless functions through invoking and UDP messages.
- Serverless workers on AWS Lambda, stand-alone running after invoked, communicate with parameter server via UDP messages.
- Memory database on ElastiCache, stores ML model, buffer space, and job configurations. The parameter server and workers exchange data via it.
- Storage service on S3, holding the training data and model result.

Before the training job starts, the training data must be split into n shards, where n is equal to the number of serverless workers and can be specified by the user. The shards of training data are stored in storage service. The workflow initiates from the parameter server, and the execution of a training job goes through the following steps:

Initialization: The parameter server gathers task parameters (data path, learning rate, thread number, an initial model, etc.), store these data in memory database. Then, it invokes n serverless functions and waits for responses from workers.

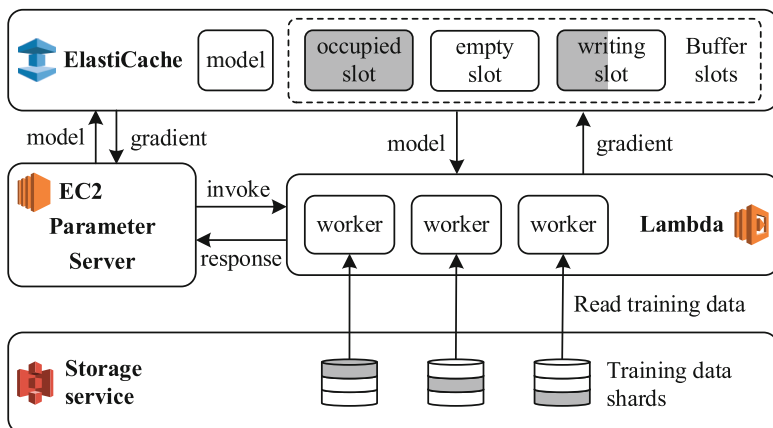


Fig. 5. Architecture overview of Chorus.

Training: When the serverless functions are started, they first read the task information and initial model from memory database. Each function reads its assigned shard and starts gradient computation.

Writing Gradient: After each iteration, the function saves the gradient to the buffer. It checks if there is available slot in the buffer. If yes, it writes the gradient data directly; if not, it uses buffer merging priority algorithm to find a suitable buffer slot and merges the gradient data into that slot. Then, it updates the task progress information in memory database and notifies the parameter server via UDP messages. According to the current synchronization model, it will choose to either block or continue to the next iteration.

Updating Model: When the parameter server receives a completion message from a serverless function via UDP, it scans all available gradients from the buffer space, reads the gradient data from a proper slot via the buffer reading priority algorithm, clears the buffer slot and updates the model parameters. Then it continues to read the next suitable buffer slot until all buffer slots are cleared. Based on the synchronization mode, it decides when to notify the serverless worker to start the next iteration.

Coordination: The parameter server checks task progress information continuously, and controls Lambda function blocking, resuming, or instance replacement via UDP communication, depending on the synchronization mode. This process continues until the training is complete, and finally the model is retrieved from ElastiCache.

Since the model is stored in memory database and the workers are independent after invoking, the only communications between parameter server and serverless workers are the invoking signal from parameter server and the finishing message from workers.

6 Evaluation

We evaluate Chorus in this section. Testing its performance from several aspects under different ML algorithms and comparing it to other system. Through these experiments, we aim to address three research questions about Chorus:

RQ1. Comparing to other synchronization modes, can the LSP mode reduce the blocking time during the training process?

RQ2. Can the LSP mode accelerates the convergence of training and save budget?

RQ3. Can the buffer system of Chorus improve performance?

6.1 Methodology

We implement Chorus on AWS. Use AWS Lambda to provide serverless service, an EC2 `t3a.medium` instance to deploy the parameter server, S3 to store the training data, and Redis database in ElastiCache to be the memory buffer space. AWS Lambda can only adjust the memory size of Lambda function (the vCPU cores will be changed along with memory size automatically). We create 10 Lambda function settings with same codes but different memory sizes, the size ranges from 768M to 3072M uniformly. All the services used and their performance information are listed in Table 1.

Table 1. Services used in the implementation of Chorus, and their performance parameters.

| Service | Performance | Price |
|-----------------------|----------------|---------------------|
| EC2 VM | t3a.medium | \$0.0376/h |
| ElastiCache (Redis 7) | cache.t3.small | \$0.034/h |
| S3 | – | – |
| Lambda function 1 | 768 MB memory | \$0.00000004167/ms |
| Lambda function 2 | 1024 MB memory | \$0.000000016667/ms |
| ... | ... | ... |

All programs are coded in Python. We implement the BSP, ASP, SSP, and LSP models, and the logistic regression and collaborative filtering algorithm.

Logistic Regression. The training of logistic regression in Chorus is implemented with stochastic gradient descent (SGD) algorithm with mini-batches. In the experiment we use Criteo dataset [16] which has a 11 GB file size and contains 45.8M entries. We split the dataset into 100 shards and upload them to S3.

Collaborative Filtering. We implement collaborative filtering algorithm based on sparse matrix decomposition and solve the decomposition with gradient descent algorithm. In the experiment we use the Netflix dataset [17] which contains 480K users' ratings over 17K films. We compress and split the dataset into 100 shards and upload them to S3.

6.2 Lambda Synchronous Parallel (LSP) Model

Blocking Time Analysis. To answer RQ1, we first use logistic regression training task as an example, the task is executed in BSP, ASP, SSP, and LSP modes, with a fixed

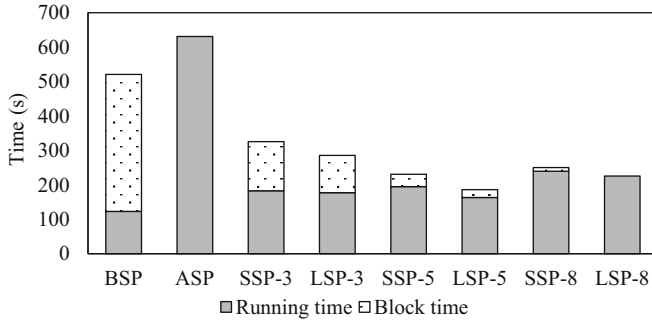


Fig. 6. Accumulated running and block time under each synchronous model

loss function value as the endpoint, and the staleness threshold takes 3, 5 and 8 values. The experiment records the total execution time and cumulative blocking time during execution. The results are shown in Fig. 6.

It can be observed that the BSP mode’s running time is long, but most of its time is spent in blocking. However, its actual running time is the shortest because BSP mode converges most stably, reaching the target loss function value in the fewest iterations. ASP mode experiences no blocking, yet its execution time is the longest due to its challenging convergence. SSP and LSP outperform BSP and ASP significantly. In all three staleness threshold configurations, LSP notably reduces system blocking time. At a threshold of 8, LSP even eliminates blocking. However, in this case, the training pace of workers becomes quite chaotic, leading to longer overall training time.

Convergence Analysis. To answer RQ2, we execute the training job of logistic regression with BSP, SSP, and LSP model. The staleness threshold of SSP and LSP are all set to 5 empirically. Figure 7 shows the results in different dimensions. Notice, in SSP and LSP model, there is boundary between epochs. So, in Fig. 7(a) and all the graphs below, the unit “Shard iteration” of the x-axis indicates the sum of iteration numbers of each shard. For instance, if one shard is calculated once, then count 1, if two shards are calculated three times each, then count 6.

Figure 7(a) shows the model converging process along with shard iterations. Although the three synchronous parallel models perform very similarly in this case, it still can be seen that the BSP mode undoubtedly has the best stability and converges the smoothest, while SSP and LSP have relatively weaker stability. Figure 7(b) shows the model convergence against time. It is observed that although the BSP model converges faster along with shard iterations, it converges slower along with time, because the blocking time between iterations is considerable. By reducing the blocking time between iterations, SSP and LSP outperform BSP, and LSP shows an even better performance than SSP.

We proceed similar experiment with the collaborative filtering algorithm, the results are shown in Fig. 8. In Fig. 8(a) the BSP model still converges the most stable. In Fig. 8(b), LSP outperforms BSP and SSP, and converges faster than the other two.

We also record the costs of these two experiments. Because AWS Lambda has a much expensive fee rate, the cost of other services is relatively ignorable, so we only

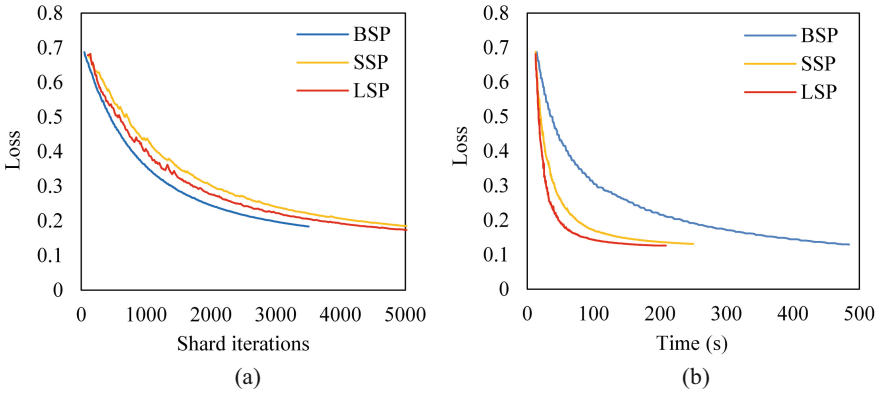


Fig. 7. Results of training Logistic Regression model.

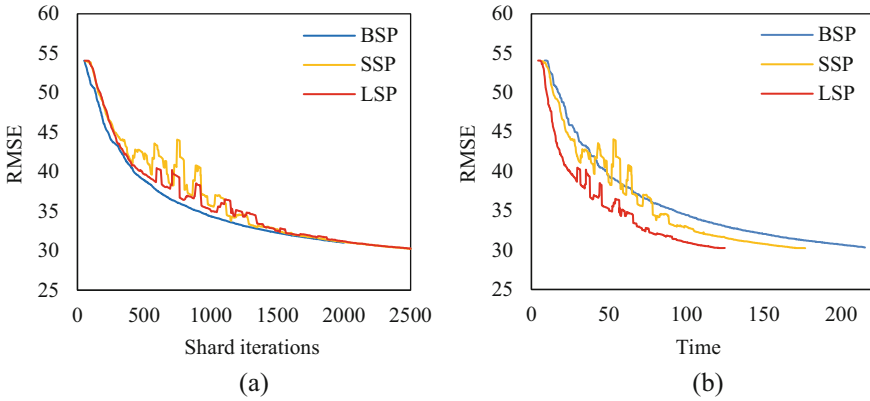


Fig. 8. Results of training Collaborative Filtering model.

list the cost of AWS Lambda here, which is shown in Table 2. In the logistic regression task, LSP respectively saves 51% and 12% costs compared to BSP and SSP, and in the collaborative filtering task, LSP saves 41% and 26% costs respectively.

We can summarize that LSP has a better performance both on convergence speed and economical cost.

Table 2. Costs of AWS Lambda in experiments.

| | BSP | SSP | LSP |
|-------------------------|--------|--------|--------|
| Logistic Regression | \$5.64 | \$3.13 | \$2.74 |
| Collaborative Filtering | \$2.92 | \$2.33 | \$1.72 |

6.3 Buffering System

To answer RQ3, we perform training of logistic regression with a fixed number of iterations. We choose the ASP model here, because ASP can give the parameter server the most pressure, the workers submit gradients continuously, giving no break time to the parameter server at all, the bottleneck effect would be extremely severe in this circumstance. We choose different groups of buffer settings, and a setting of no buffer system, the workers transmit data with parameter server directly.

Figure 9(a) shows the convergence process during training, and Fig. 9(b) shows the total time costs of the experiments. In Fig. 9(a), the black curve is shorter because when there is no buffer system, the gradient data are transferred serial, but the loss value of model can only be calculated after all shards are calculated at least once. Therefore, before the gradients of the first batch are all received, the parameter server cannot calculate any meaningful loss value.

We can see the buffer system accelerates the training process dramatically. Even with only one buffer slot, the convergence is still faster several times than training with no buffer system. According to the experimental result in Fig. 9(b), compared to no buffer mode, the buffer system accelerates the training job from $4.7\times$ (slot = 1) to $8.4\times$ (slot = 5).

The buffering system disperses the burden of merging gradients among Lambda functions, resulting in greater adaptability to task scaling. As the task scale increases, users only need to expand a few buffer slots (in ElastiCache) to accommodate it, without adding expensive VMs. Most of the computational tasks are dispersed among Lambda functions, which are paid by usage.

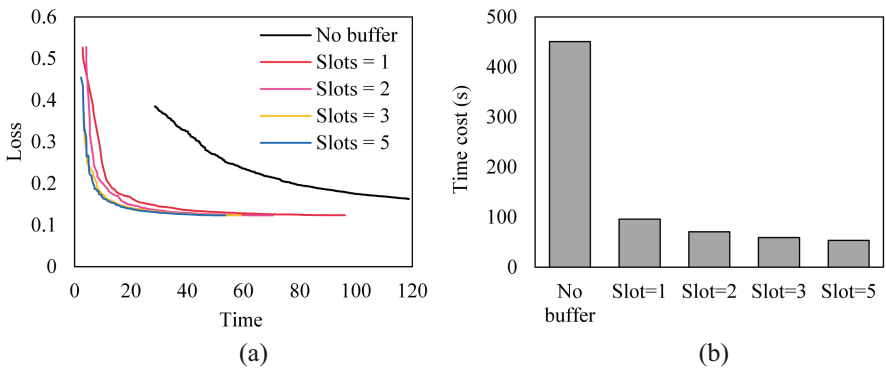


Fig. 9. Training results with different buffer settings.

6.4 Comparison

To answer RQ3, we selected Berkeley's serverless machine learning system Cirrus [10] as target. Cirrus is a typical open-source serverless framework dedicate to machine learning, it is also implemented on AWS Lambda, which providing a relatively fair platform

for comparison. Cirrus also uses a parameter server model, and similarly implements logistic regression and collaborative filtering algorithms, making it easy to compare horizontally. In this experiment, Cirrus' parameter server is deployed on the same AWS EC2 `t3a.medium` instance, using 100 Lambda functions for collaboration. Chorus uses the LSP model with staleness threshold set to 5 and 10 buffer slots. Training is conducted separately for the logistic regression and collaborative filtering algorithms, and the change in loss function is recorded. The results obtained are shown in Fig. 10. From the experimental results, Cirrus based on the BSP mode has a smoother convergence curve, but Chorus shows a faster convergence speed, and thus result in a lower cost.

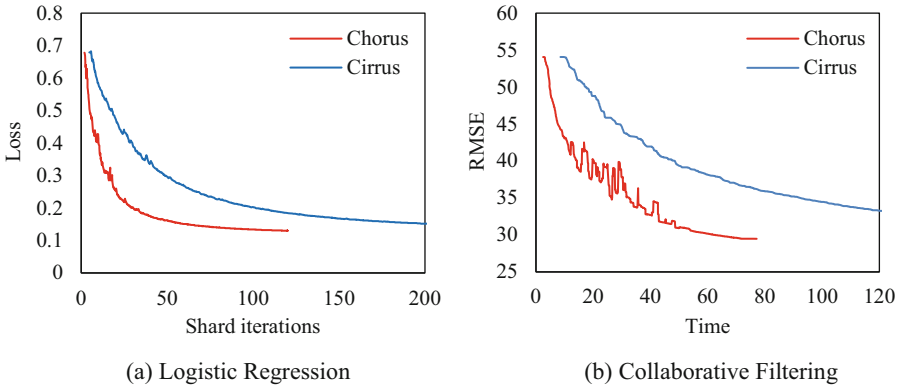


Fig. 10. Comparisons between the training results of Chorus and Cirrus

7 Conclusion

This work introduces Chorus, a distributed machine learning framework on serverless platform. We propose the novel Lambda Synchronous Parallel (LSP) model in Chorus. LSP is an upgrade of SSP, it adjusts resource allocation of functions dynamically during training process, and skew the resource towards lagging workers, thus reduce blocking time naturally, and making the model convergence faster and more economically. We design a buffer system in Chorus, to deal with the bottleneck problem on the parameter server. We set multiple buffer slots to save temporary gradients, the parameter server and workers exchange data via the buffer slot, thus alleviate the network bottleneck problem. To alleviate the computational bottleneck problem of parameter server, we design buffer merging mechanism, and propose specific buffer merging and reading priority algorithms to avoid training imbalance. Through the buffer system, we actually disperse computational job of parameter server among Lambda workers, thus lighten the burden of the parameter server significantly.

Chorus is implemented on AWS platform. In the experiments and performance inspection from several aspects, LSP shows a better convergence efficiency than BSP and SSP model. The buffering mechanism also shows impressive performance improvements, accelerating the training job from 4.7x to 8.4x. Chorus also outperforms Cirrus in equivalent experiments.

Acknowledgments. This work is partially supported by National Natural Science Foundation of China (No. 42361144884, No. 61972386).

References

1. Jonas, E., Pu, Q., Venkataraman, S., Stoica, I., Recht, B.: Occupy the cloud: distributed computing for the 99%. In: Proceedings of the 2017 Symposium on Cloud Computing, pp. 445–451 (2017)
2. Carver, B., Zhang, J., Wang, A., Cheng, Y.: In search of a fast and efficient serverless dag engine. In: 2019 IEEE/ACM Fourth International Parallel Data Systems Workshop (PDSW), pp. 1–10. IEEE (2019)
3. Shankar, V., et al.: Numpywren: serverless linear algebra. arXiv preprint [arXiv:1810.09679](https://arxiv.org/abs/1810.09679) (2018)
4. Fouladi, S., et al.: Encoding, fast and slow: low-latency video processing using thousands of tiny threads. In: NSDI, vol. 17, pp. 363–376 (2017)
5. Ao, L., Izhikevich, L., Voelker, G.M., Porter, G.: Sprocket: a serverless video processing framework. In: Proceedings of the ACM Symposium on Cloud Computing, pp. 263–274 (2018)
6. Fouladi, S., Romero, F., Iter, D., Li, Q., Chatterjee, S.: From laptop to lambda: outsourcing everyday jobs to thousands of transient functional containers. In: 2019 USENIX Annual Technical Conference (USENIX ATC 2019), p. 14 (2019)
7. Li, M., et al.: Scaling distributed machine learning with the parameter server. In: 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2014), pp. 583–598 (2014)
8. Ho, Q., et al.: More effective distributed ML via a stale synchronous parallel parameter server. In: Advances in Neural Information Processing Systems, vol. 26 (2013)
9. Aytekin, A., Johansson, M.: Harnessing the power of serverless runtimes for large-scale optimization. [arXiv:1901.03161](https://arxiv.org/abs/1901.03161) (2019)
10. Carreira, J., Fonseca, P., Tumanov, A., Zhang, A., Katz, R.: Cirrus: a serverless framework for end-to-end ML workflows. In: Proceedings of the ACM Symposium on Cloud Computing, pp. 13–24 (2019)
11. Wang, H., Niu, D., Li, B.: Distributed machine learning with a serverless architecture. In: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, pp. 1288–1296. IEEE (2019)
12. Ali, A., Zawad, S., Aditya, P., Akkus, I.E., Chen, R., Yan, F.: SMLT: a serverless framework for scalable and adaptive machine learning design and training. [arXiv:2205.01853](https://arxiv.org/abs/2205.01853) (2022)
13. Gimeno Sarroca, P., Sánchez-Artigas, M.: MLLess: achieving cost efficiency in serverless machine learning training. *J. Parallel Distrib. Comput.* **183**, 104764 (2024)
14. Feng, L., Kudva, P., Da Silva, D., Hu, J.: Exploring serverless computing for neural network training. In: 2018 IEEE 11th International Conference on Cloud Computing (CLOUD) (2018)
15. Thorpe, J., et al.: Dorylus: affordable, scalable, and accurate GNN training with distributed CPU servers and serverless threads. In: 15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2021), pp. 495–514 (2021)
16. Criteo_dataset | Kaggle. <https://www.kaggle.com/datasets/mrkmakr/criteo-dataset>. Accessed 1 Apr 2024
17. Netflix Prize data | Kaggle. <https://www.kaggle.com/netflix-inc/netflix-prize-data>. Accessed 1 Apr 2024



Evaluating Performance of LLaMA2 Large Language Model Enhanced by QLoRA Fine-Tuning for English Grammatical Error Correction

Jing An¹, Yanbing Bai^{2(✉)}, Jiye Li³, Junjie Hu⁴, Rui Li¹, Yuxi Xiao¹,
and Rui Hua¹

¹ School of Translation and Interpreting, Beijing International Studies University,
Beijing 100024, China

² Center for Applied Statistics, School of Statistics, Renmin University of China,
Beijing 100872, China
ybbai@ruc.edu.cn

³ Department of Computer Science and Engineering, University of Yamanashi,
Kofu 4008511, Japan

⁴ Shenzhen Institute of Artificial Intelligence and Robotics for Society,
Shenzhen 518116, China

Abstract. Large Language Models (LLMs) have experienced significant advancements across various contexts. However, their impact on vertical fields remains understudied and unsatisfactory due to the heightened requirement for domain-specific expertise in these fields. English Grammar Error Correction (GEC) is urgently needed in the current academic and educational fields, which are currently full of challenges regarding precision, adaptability, and complex grammatical mistakes. The release of the C4_200M Synthetic Dataset and advancements in LLaMA2's QLoRA fine-tuning technology present an unprecedented opportunity to examine these issues more closely. This study aims to assess the performance of the LLaMA2 in the area of GEC. In this study, we implemented LLaMA2 augmented with QLoRA finetune model in Spark scalable cluster processing environment, and we investigated model performance under two methods, Zero-shot and Few-shot prompting, and configured the parameters for text generation, including Top-p, Top-k, and Beam search. We built an efficient and accurate scalable system, with BLEU from 12.33 to 14.8, ROUGE from 19.33% to 25.97% and the editing distance from 4.21 to 1.89, providing a solid foundation for future work. The code of this paper is available at [LINK](#).

Keywords: Large Language Model · QLoRA · Grammatical Error Correction · Spark

1 Introduction

Natural Language Processing (NLP) has witnessed substantial advancements, highlighted by Facebook's unveiling of the Llama series, including the original

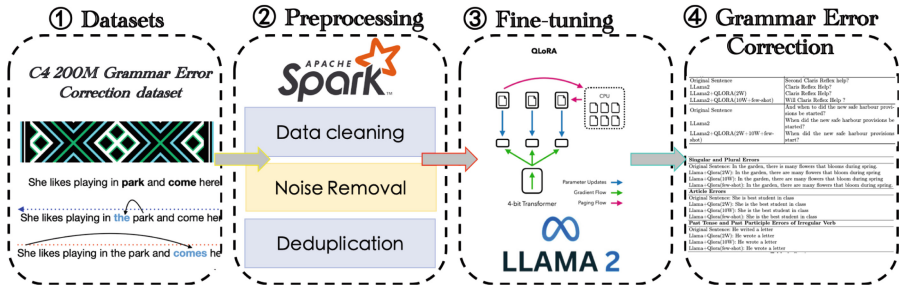


Fig. 1. The framework of our method.

LLaMA [1] and the more advanced LLaMA2 [2]. These models are important in academic and professional communities due to their superior performance across a range of applications, such as financial news analysis [3], medical diagnostics [4], and online sexual predatory chat detection [5] (Fig. 1).

Despite the versatility of large-scale models in various scenarios, optimizing their effectiveness for specific applications meets with many difficulties, particularly in the educational industry. Text as a pivotal medium for expressing messages and communication become more and more important, driving the need for more sophisticated English text error correction technologies. The domain of English Grammar Error Correction (GEC), particularly with Large Language Models (LLMs), has not been investigated rigorously. There are also many challenges in Grammar Error Correction, especially concerning precision, adaptability, and complex grammatical mistakes.

The introduction of the C4_200M Synthetic Dataset for Grammatical Error Correction [6], combined with advancements in LLaMA2’s QLoRA [7] fine-tuning technology, presents an unprecedented opportunity to examine these issues more closely. This study is propelled by such innovations and seeks to assess the performance of the LLaMA2 Large Language Model in the area of GEE.

The contributions of this research are as follows.

- This study offers a highly efficient, flexible solution for English grammatical error corrections by utilizing the Spark distributed computing platform.
- We establish a benchmark for subsequent research in this domain. This is the first study that employs the LLaMA2 model with parameter-efficient fine-tuning for the correction of English text errors to the best of our knowledge.
- We conduct an in-depth case analysis of English grammar correction errors to discern the root causes of these inaccuracies. This study offers some insightful recommendations for future model optimizations.

2 Related Work

2.1 Grammar Error Correction

Recent research on Grammar Error Correction (GEC) has predominantly centered on its application within specific sectors. Fan et al. introduced Grammar GPT, an open-source large language model (LLM) designed to investigate its capabilities for correcting grammatical errors in native Chinese texts [8]. Pantazis et al. developed RUSTASSISTANT, a tool leveraging LLMs to propose corrections for Rust compilation errors [9]. Song et al. established two methodologies utilizing LLMs to assess GPT-4’s proficiency in explaining grammar errors [10]. Christopher et al. explored both open-source and commercial language models for grammatical error correction in texts written by English learners [11]. Maria et al. assessed the performance of GPT-3.5 and GPT-4 in GEC tasks for Brazilian Portuguese [12]. Zhang et al. employed RobustGEC to demonstrate that contemporary GEC systems still struggle to maintain robustness amidst contextual disturbances [13]. However, these investigations have been constrained by their focus on niche domains and reliance on preceding technologies, with their scope further limited by the small sizes of their sample sets. Traditionally, common benchmark datasets have been utilized to evaluate GEC effectiveness.

Furthermore, the GEC research landscape in recent years has shifted towards enhancing model accuracy through the adoption of novel fine-tuning techniques [3, 5, 8, 9, 14–16]. Thus, investigating the latest fine-tuning technologies and leveraging extensive benchmark GEC datasets holds substantial importance. The unveiling of the C4.200M Synthetic Dataset for Grammatical Error Correction [6], coupled with the advancements in Llama2’s QLoRA fine-tuning technology [7], offers a unique opportunity to delve deeper into these challenges. Motivated by these developments, this study aims to employ QLoRA fine-tuning technology to evaluate the C4 dataset, marking a significant step forward in the field of GEC.

2.2 LLaMA2 Large Language Model

LLaMA2 [2], an open-source large language model developed by the Facebook team, whose powerful language understanding capabilities, efficient processing speed, and excellent generalization abilities constitute its core advantages comparable to ChatGPT. LLaMA2 introduces advanced positional encoding called RoPE (Rotary Position Embedding), which provides an effective way for the model to capture the positional relationships of words in a sequence, thereby enhancing the model’s understanding of text structure. Additionally, the model utilizes RMSNorm and optimized SwiGLU activation functions, significantly improving the speed and efficiency of handling large amounts of text. The structural design of LLaMA2 reflects its efficiency and flexibility in processing. In the input stage of the model, text is first tokenized, converting natural language into a numerical form that the model can understand. These numerical representations are then passed through an embedding layer to transform them into

corresponding vectors, forming the backbone of the model, which consists of several layers of Transformer networks responsible for outputting hidden states, i.e., hidden state vectors, carrying deep semantic information of the text. In summary, LLaMA2, as an advanced large language model, embodies the forefront of current natural language processing technology in its structural design and principles. Through its refined structure design and multi-task adaptability, LLaMA2 can provide efficient and accurate solutions for various language processing tasks, making it a model worth attention in the current natural language processing field.

3 Methodology

3.1 Data Preparation

The C4_200M Synthetic Dataset for Grammatical Error Correction was utilized in this study. This dataset contains up to 185 million sentence pairs, each consisting of one grammatically correct sentence and one error-added sentence, suitable for models of Grammar error correction and language understanding. This dataset contains a diverse range of textual content, such as news articles, blog posts, etc. This diversified content of the C4 dataset provides a wide range of linguistic contexts and linguistic styles for the grammar correction model. The data preprocessing was implemented in the Spark Cluster computing environment to achieve scalable and efficient processing. The data processing stage mainly includes deleting all irrelevant characters other than English, replacing the comma with a special tag, and converting all characters to lowercase. These preprocessed data will be used as training data for subsequent model fine-tuning and training.

The quality, type, and amount of data have an important impact on the accuracy of fine-tuning the model [17–19]. In general, optimizing the three dimensions of quantity, quality, and diversity can not only enhance the effectiveness of model learning but also ensure that the model can achieve the best performance in various tasks. To build high-quality datasets for fine-tuned modeling, We utilized Locality-Sensitive Hashing (LSH) to implement the text deduplication algorithm. In the data cleaning and deduplication phase, the introduction of Spark greatly enhances the processing power. By converting a custom hash function into a user-defined function (UDF) and applying it to a Spark DataFrame, parallel hash processing is implemented on each node of the cluster. In addition, Spark effectively identifies and removes duplicates by grouping and aggregating methods and by calculating Hamming distances between data items in the hash bucket.

By combining the locally sensitive hashing and Spark’s distributed computing capabilities, it not only optimizes the efficiency of the algorithm side and improves the speed of data processing, but also ensures the high quality and diversity of the dataset. Besides, it demonstrates the importance and feasibility of efficient and accurate data processing in the face of large-scale high-dimensional datasets.

To improve the diversity of fine-tuning data and optimize the model training effect, a strategy combining the BERT model, cluster analysis, and hierarchical

random sampling is designed. We first perform text cleaning, fill truncation, word segmentation, labeling, and encoding. Then, using the pre-trained BERT model provided by HuggingFace, we extract the embedding vector of the text data through the last hidden layer of the model. After average pooling, these vectors are formed into fixed-length vectors, which can effectively capture the complex semantic information of the input text. Next, we use PCA and t-SNE methods to reduce the dimension of these embedding vectors and analyze the distribution of the vectors by visual interpretation. The observed distribution shows that these vectors can be efficiently grouped based on clustering algorithms. Moreover, the analysis of cluster analysis results shows that sentences in the same cluster often have similar emotions, contexts, or the same keywords, which verifies the effectiveness of cluster classification. Finally, based on the clustering results, we implement a stratified random sampling strategy to evenly extract samples from each cluster and compose the final fine-tuned data set. This approach not only ensures a representative and diverse dataset but also provides a balanced and informative database for model fine-tuning.

3.2 Prompts and Learning Strategies

To ensure the efficiency and effectiveness of model training, we adopt a format that mainly consists of three key components: instructions, inputs, and outputs. Firstly, the instruction section clearly defines the operations or task constraints that the model needs to follow, providing the context for the model to execute the task. The subsequent input section provides specific data information corresponding to the instructions, supplementing the details of the instructions to help the model understand the specific requirements of the task. Finally, the output section specifies the expected results that the model should produce under given instructions and input conditions, i.e., correct labels or answers.

In this work, we adopted Zero-shot and Few-shot learning strategies to evaluate the performance and adaptability of the model under different learning conditions. The Zero-shot learning strategy aims to test the model’s ability to handle new tasks without any previous examples or guidance, relying on the model’s generalization ability to make predictions. For this purpose, we designed examples containing descriptive task instructions and contextual inputs that aim to evaluate the model’s understanding and task execution ability without specific training. An example of Zero-shot learning is as follows: *“Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.”*

In contrast, the few-shot learning strategy guides the model to learn how to handle specific tasks by providing a small number of selected examples. In this strategy, we particularly focus on the model’s ability to identify and correct common English grammar and spelling errors. By providing a series of sentences containing typical errors and their correct forms, the model receives direct guidance on how to recognize and correct these errors. These examples cover common grammar errors such as subject-verb disagreement, incorrect tense usage, inappropriate preposition usage, as well as spelling errors. An example of Few-shot

learning is as follows: “Please correct any grammar, spelling, or punctuation errors in the following English sentences.”

| <i>Incorrect</i> | <i>Correct</i> |
|---|--|
| <i>He never have gone to an overseas country</i> | <i>He has never gone to an overseas country</i> |
| <i>There’s many reasons why I don’t agree with you</i> | <i>There are many reasons why I don’t agree with you</i> |
| <i>She written a book about her adventures in Asia</i> | <i>She wrote a book about her adventures in Asia</i> |
| <i>The cats tail was black and white</i> | <i>The cat’s tail was black and white</i> |
| <i>Its important to be consistent in you’re studies</i> | <i>It’s important to be consistent in your studies</i> |

Through Zero-shot learning, we assess the model’s ability to learn autonomously when facing completely unknown tasks, while through Few-shot learning, we test the model’s ability to adapt and learn quickly using a small number of examples. The combination of these two learning strategies allows us to comprehensively understand the model’s performance at different levels of guidance, especially in terms of accuracy and efficiency in handling English grammar and spelling errors.

3.3 Parameter Efficient Fine-Tuning via QLoRA

In the initial evaluation of this study, LLaMA2 was used directly to identify grammatical errors in 500 sampled sentences with only 30% accuracy. Therefore, we developed a detailed plan to download and use the pre-trained LLaMA2 model and fine-tune the data set for the specific scenario. The implementation of this scheme will greatly improve the performance of the system and the accuracy of error correction, and then meet the needs of users.

QLoRA is an innovative technology that combines the idea of Low-Rank Adaptation (LoRA) with advanced quantitative compression techniques to improve the efficiency and effectiveness of fine-tuning large-scale pre-trained language models (PLMs). This approach recognizes that despite the large model’s large number of parameters, the performance of the model depends primarily on the content of the low-rank dimension, that is, the model has a small intrinsic dimension.

Based on the inherent low-rank characteristics of the large model, the bypass matrix is added to simulate the parameter fine-tuning of the whole model. LoRA achieves the purpose of lightweight fine-tuning through a simple and effective scheme. The idea of LoRA is to add a bypass next to the original PLM to do a dimensionality reduction and then a dimensionality enhancement operation to simulate the intrinsic rank. During training, the parameters of PLM are fixed, and only dimensionality reduction matrix A and dimensionality increase matrix B are

trained. While the input and output dimensions of the model remain unchanged, the parameters of BA and PLM are superimposed when output is performed. Initialize A with a random Gaussian distribution and initialize B with a zero matrix to ensure that the bypass matrix is still a zero matrix at the beginning of training.

In addition, Parameter Efficient fine-tuning also utilizes NF4 quantization compression technology to describe parameters using low-dimensional video memory according to weight distribution laws, reducing data storage requirements while maintaining or even better than the original accuracy during BF16 calculations. This approach not only optimizes the storage and computational efficiency of the model but also ensures that data quality is more important than data volume during the fine-tuning process.

By combining the fine-tuning concept of LoRA with NF4’s quantitative compression technology, QLoRA fine-tuning provides a lightweight and efficient fine-tuning solution for large models. It allows for good fine-tuning results with fewer training resources, making it possible for various large models to be transformed into specialized models adapted to the needs of different domains after light fine-tuning. This fine-tuning method not only enhances the applicability of the model, but also greatly improves the efficiency of fine-tuning, and opens up new possibilities for the application of large models.

4 Experiments

4.1 Text Generation Settings

We first introduce the text generation settings when fine-tuning LLaMA2. We set the temperature to 0.1 to increase the certainty of the model output; Top-p is set to 0.75, ensuring that the model only considers words with a cumulative probability of at least 75%; The Top-k limit is 40, allowing the model to consider only the most likely 40 words per step. We also use a four-beam Beam search, which helps improve the text quality. Finally, the maximum length of the generated text is set to the context length plus five. These parameters work together to optimize the quality and relevance of the generated text.

Overall, this set of parameter configurations is designed to optimize the LLaMA model’s performance on syntax-correcting tasks by precisely controlling the generation process. By adjusting the temperature, Top-p, Top-k, and Beam search parameters, the model generates more accurate, consistent, and high-quality syntax corrections, improving overall task efficiency and accuracy.

4.2 Parameter Settings of Training

```
Python
max_length = 512
device_map = "auto"
batch_size = 128
micro_batch_size = 32
```

Table 1. Model evaluation index records.

| | LLaMA2 | LLaMA2+QLoRA (2W) | LLaMA2+QLoRA (10W) | LLaMA2+QLoRA (few-shot) |
|------------------|--------|----------------------|-----------------------|----------------------------|
| BLUE | 12.23 | 14.33 | 14.83 | 14.80 |
| ROUGE | 19.33 | 23.02 | 25.32 | 25.97 |
| Editing Distance | 4.21 | 2.31 | 2.00 | 1.89 |

```
gradient_accumulation_steps = batch_size // micro_batch_size
```

```
bnb_config = BitsAndBytesConfig
    (load_in_4bit=True, # load the model into memory using
    4-bit precision\
    bnb_4bit_use_double_quant=True, # use double quantition
    bnb_4bit_quant_type="nf4", # use NormalFloat quantition
    bnb_4bit_compute_dtype=torch.bfloat16 # use hf for computing
    when we need)
```

```
The parameters for the model training are as above.\
```

We tried fine-tuning different data volumes, including 20000 and 100000.

4.3 Experimental Results

In the results analysis, we use three metrics to evaluate the performance of the models on text correction tasks, including BLEU, ROUGE, and editing distance. Table 1 lists the results.

BLEU evaluates the similarity between the generated text and the target text (i.e., the correct answer) using an n-gram match. In the syntax error correction task, if the model output is more similar to the correct answer, the error correction effect is better. The LLaMA2 model scores 12.23 on this metric, providing us with a baseline.

For the ROUGE, it measures the ability of the generated text to retain the main meaning of the original sentence. For error correction tasks, this means that the core information of the original sentence is preserved while the error is corrected. The original model gets 19.33% on this metric.

We also focus on editing distance, which measures the difference between the model's output and the correct answer. In text correction, the lower editing distance indicates that the model makes more accurate changes and the outputs are closer to the correct answer.

After the introduction of QLoRA technology, there was a significant improvement in all indicators. After 20,000 training iterations, the BLEU score improved to 14.33, the ROUGE score reached 23.02%, and the editing distance dropped to 2.31. When the training iteration increased to 100,000 times, these indicators further improved, reflecting the continuous enhancement of the model performance.

Table 2. Result examples.

| | |
|--------------------------------|---|
| Original Sentence | At that stage, the clinical implication with our finds is unknow |
| LLaMA2 | At that stage, the clinical implication with our finds are unknow |
| LLaMA2+QLoRA (2W+10W+few-shot) | At that stage, the clinical implication of our finds are unknow |
| Original Sentence | Alice said that she was going to the park the next day |
| LLaMA2 | Alice said that she was going to the park the next day |
| LLaMA2+QLoRA (2W+10W+few-shot) | Alice said that she is going to the park the next day |
| Original Sentence | Second Claris Reflex help? |
| LLaMA2 | Claris Reflex Help? |
| LLaMA2+QLoRA (2W) | Claris Reflex Help? |
| LLaMA2+QLoRA (10W+few-shot) | Will Claris Reflex Help ? |
| Original Sentence | And when to did the new safe harbour provisions be started? |
| LLaMA2 | When did the new safe harbour provisions be started? |
| LLaMA2+QLoRA (2W+10W+few-shot) | When did the new safe harbour provisions start? |

Finally, the few-shot learning method was used to further optimize the model. This approach helps models to adapt to specific tasks by providing a small number of concrete examples. The results show that Few-shot learning significantly improves the model’s performance on all three key metrics, with BLEU scoring 14.80, ROUGE scoring 25.97%, and editing distance decreasing to 1.89. These results highlight the importance of few-shot learning in improving the adaptability of models to new tasks.

Table 2 shows some examples of the results. It can be seen from the model results that the fine-tuned model can have better grammar error correction ability than the original LLaMA model, and the model’s error correction ability is better than that of the original LLaMA model on medium-difficulty sentences. In addition, LLaMA models that used more samples for fine-tuning and used few-shot showed better error correction than LLaMA models with fewer samples and zero-shot.

4.4 Error Type Analysis

To better analyze the syntax correction ability of the model, several syntax error types are analyzed on the llama model individually. Judging from the results of

Table 3. Description of the developed models, divided into three categories: *Singular and Plural*, *Article*, *Past Tense and Past Participle of Irregular Verb*. All the models can correct these error correction.

| |
|--|
| Singular and Plural Errors |
| Original Sentence: In the garden, there is many flowers that blooms during spring |
| LLaMA+QLoRA (2W): In the garden, there are many flowers that bloom during spring |
| LLaMA+QLoRA (10W): In the garden, there are many flowers that bloom during spring |
| LLaMA+QLoRA (few-shot): In the garden, there are many flowers that bloom during spring |
| Article Errors |
| Original Sentence: She is best student in class |
| LLaMA+QLoRA (2W): She is the best student in class |
| LLaMA+QLoRA (10W): She is the best student in class |
| LLaMA+QLoRA (few-shot): She is the best student in class |
| Past Tense and Past Participle Errors of Irregular Verb |
| Original Sentence: He writed a letter |
| LLaMA+QLoRA (2W): He wrote a letter |
| LLaMA+QLoRA (10W): He wrote a letter |
| LLaMA+QLoRA (few-shot): He wrote a letter |

the model, the model excels at simple error correction problems, such as singular and plural errors, article errors, and past tense and past participle errors of irregular verbs. All the models can correct these error corrections. Table 3 shows some examples.

However, for moderate-difficulty error correction, such as tense errors, passive voice errors, and preposition errors, the error correction ability of large-sample is better than that of small-sample and zero-shot models. Table 4 lists some examples. One possible explanation for this is that this type of error correction is relatively rare, and the model has the corresponding learning ability. The corresponding abilities can be effectively stimulated by few-shot.

In addition, for more difficult error correction tasks, such as non-predicate verb errors and conversion errors between direct and indirect speech, all the models cannot correctly modify these errors. Table 5 lists some examples. It is believed that models can not well understand the text and what these error types mean, these models just correct those sentences separately and ignore the logical relationship between sentences. In the above sentences, apart from the given errors, there is no other error in every signal sentence. Therefore, models may not have high-level grammar knowledge, and further learning is needed.

Table 4. Description of the developed models into *Tense errors*.

| Tense errors |
|---|
| Original Sentence: When I saw him last, he told me that he looks forward to visit his family |
| Llama+Qlora (2W): When I saw him last, he told me that he looks forward to visit his family |
| Llama+Qlora (10W): When I saw him last, he told me that he was looking forward to visit his family |
| Llama+Qlora (few-shot): When I saw him last, he told me that he looking forward to visit his family |
| Correct Answer: When I saw him last, he told me that he was looking forward to visiting his family |

4.5 Performance Evaluation of Open Source Language Tools

To further evaluate the output of the model, we tried to use further open-source language tools of Language Tool and Grammarly to analyze the output of the model. Language Tool is an open-source syntax and spelling checking tool that helps users check for errors and irregular usage in text. It can support many languages, including English, German, French, Spanish, Russian, etc. It can identify error types including grammatical errors, spelling errors, punctuation errors, inconsistent styles, repetitive words, etc. However it cannot identify the deep meaning and context understanding of the text.

The analysis results of Language Tool show that the Llama model based on Lora fine-tuning can effectively reduce errors in sentences, with the best effect on grammatical errors, but poor identification and modification of punctuation errors. At the same time, we noticed that Language Tool does not identify errors in the sentence well, so we used Grammarly to make further analysis of the output of the model.

Grammarly is a popular online grammar-checking, spelling-checking, and text-proofreading tool. It is also able to check for spelling and grammatical errors, as well as punctuation errors, sentence structure problems, style inconsistency, and so on. At the same time, its spelling check is not only based on the dictionary but also considers the context to provide more accurate advice. Moreover, the advanced version of Grammarly identifies more complex grammatical problems such as passive voice, long sentence structure, etc. Therefore, its identification results are more accurate than those of Language Tool results.

The analysis results are shown in the following figure. We can see that the LLaMA model based on LoRA fine-tuning can effectively reduce errors in sentences, with the best effect on grammatical errors, but poor identification and modification of punctuation errors. At the same time, we noticed that there are still errors in the correct sentences in the data.

Table 5. Description of the developed models into two categories: *Conjunction* and *Conversion between direct and indirect speech*.

| Conjunction Errors |
|---|
| Original Sentence: She wanted to go to the movie, or she wanted to stay home, and she decided to cook dinner, but she didn't have all the ingredients, so she went out anyway |
| LLaMA+QLoRA (2W): She wanted to go to the movies, or she wanted to stay home, and she decided to cook dinner, but she didn't have all the ingredients, so she went out anyway |
| LLaMA+QLoRA (10W): She wanted to go to the movies, or she wanted to stay home, and she decided to cook dinner, but she didn't have all the ingredients, so she went out anyway |
| LLaMA+QLoRA (few-shot): She wanted to go to the movies, or she wanted to stay home, and she decided to cook dinner, but she didn't have all the ingredients, so she went out anyway |
| Correct Answer: She couldn't decide whether to go to the movie or stay home, but since she didn't have all the ingredients to cook dinner, she went out anyway |
| Conversion errors between direct and indirect speech |
| Original Sentence: She said, 'I am tired.' He said she is tired |
| LLaMA+QLoRA (2W): She said, "I am tired." He said, "You are tired." |
| LLaMA+QLoRA (10W): She said, "I am tired." He said, "You are tired." |
| LLaMA+QLoRA (few-shot): She said, "I am tired." He said, "You are tired." |
| Correct Answer: She said, 'I am tired.' He said she was tired |

5 Conclusion

In this paper, we developed an English text error correction system using Spark and LLaMA2 models, fine-tuned with a C4 syntax correction training set. This model effectively corrected grammatical and spelling errors in sentences, aligning with our expectations. Our work highlights the potential of large models in enhancing language learning and teaching quality. By providing accurate grammar error correction and real-time feedback, our system aids learners in mastering English grammar, thereby improving language learning efficiency. However, our model faces limitations, including dataset quality issues and hardware constraints. Future efforts will focus on improving dataset quality, ensuring original sentence correctness, and enhancing hardware support to boost model performance.

Acknowledgments. This work was jointly supported by National Natural Science Foundation of China (NSFC) under grants 62206301; Public Health & Disease Control and Prevention, Fund for Building World-Class Universities (Disciplines) of Renmin University of China. Project No. 2024PDPC; the Major Project of the MOE (China)

National Key Research Bases for Humanities and Social Sciences (22JJD910003); Wine Group's research grant No. 09202188. This work was supported by Public Computing Cloud, Renmin University of China. We sincerely thank the students at Renmin University of China for providing data processing and experiment support.

References

1. Touvron, H., Lavril, T., Izacard, G., et al.: LLaMA: open and efficient foundation language models. arXiv preprint [arXiv:2302.13971](https://arxiv.org/abs/2302.13971) (2023)
2. Touvron, H., Martin, L., Stone, K., et al.: Llama 2: open foundation and fine-tuned chat models. arXiv preprint [arXiv:2307.09288](https://arxiv.org/abs/2307.09288) (2023)
3. Pavlyshenko, B.M.: Financial news analytics using fine-tuned Llama 2 GPT model. arXiv preprint [arXiv:2308.13032](https://arxiv.org/abs/2308.13032) (2023)
4. Zhao, H., et al.: Ophtha-LLaMA2: a large language model for ophthalmology. arXiv preprint [arXiv:2312.04906](https://arxiv.org/abs/2312.04906) (2023)
5. Nguyen, T.T., et al.: Fine-tuning Llama 2 large language models for detecting online sexual predatory chats and abusive texts. arXiv preprint [arXiv:2308.14683](https://arxiv.org/abs/2308.14683) (2023)
6. Stahlberg, F., Kumar, S.: Synthetic data generation for grammatical error correction with tagged corruption models. arXiv preprint [arXiv:2105.13318](https://arxiv.org/abs/2105.13318) (2021)
7. Dettmers, T., et al.: QLoRA: efficient finetuning of quantized LLMs. arXiv preprint [arXiv:2305.14314](https://arxiv.org/abs/2305.14314) (2024)
8. Fan, Y., et al.: GrammarGPT: exploring open-source LLMs for native Chinese grammatical error correction with supervised fine-tuning. arXiv preprint [arXiv:2307.13923](https://arxiv.org/abs/2307.13923) (2023)
9. Deligiannis, P., et al.: Fixing rust compilation errors using LLMs. arXiv preprint [arXiv:2308.05177](https://arxiv.org/abs/2308.05177) (2023)
10. Song, Y., et al.: GEE! Grammar error explanation with large language models. arXiv preprint [arXiv:2311.09517](https://arxiv.org/abs/2311.09517) (2023)
11. Davis, C., et al.: Prompting open-source and commercial language models for grammatical error correction of English learner text. arXiv preprint [arXiv:2401.07702](https://arxiv.org/abs/2401.07702) (2024)
12. Penteado, M.C., Perez, F.: Evaluating GPT-3.5 and GPT-4 on grammatical error correction for Brazilian Portuguese. arXiv preprint [arXiv:2306.15788](https://arxiv.org/abs/2306.15788) (2023)
13. Zhang, Y., et al.: RobustGEC: robust grammatical error correction against subtle context perturbation. arXiv preprint [arXiv:2310.07299](https://arxiv.org/abs/2310.07299) (2023)
14. Yang, C.-H.H., Gu, Y., Liu, Y.-C., Ghosh, S., Bulyko, I., Stolcke, A.: Generative speech recognition error correction with large language models and task-activating prompting. arXiv preprint [arXiv:2309.15649](https://arxiv.org/abs/2309.15649) (2023)
15. Kaneko, M., Okazaki, N.: Controlled generation with prompt insertion for natural language explanations in grammatical error correction. arXiv preprint [arXiv:2309.11439](https://arxiv.org/abs/2309.11439) (2023)
16. Kaddour, J., Liu, Q.: Text data augmentation in low-resource settings via fine-tuning of large language models. arXiv preprint [arXiv:2310.01119](https://arxiv.org/abs/2310.01119) (2023)
17. Ji, Y., et al.: Exploring the impact of instruction data scaling on large language models: an empirical study on real-world use cases. arXiv preprint [arXiv:2303.14742](https://arxiv.org/abs/2303.14742) (2023)
18. Chen, H., et al.: Maybe only 0.5% data is needed: a preliminary exploration of low training data instruction tuning. arXiv preprint [arXiv:2305.09246](https://arxiv.org/abs/2305.09246) (2023)
19. Zhou, C., et al.: LIMA: less is more for alignment. In: Thirty-Seventh Conference on Neural Information Processing Systems (2023)



A Label Embedding Algorithm Based on Maximizing Normalized Cross-Covariance Operator

Yulin Xue, Yuchen Pan, Tao Peng, Jun Li, and Jianhua Xu^(✉)

School of Computer and Electronic Information, School of Artificial Intelligence,
Nanjing Normal University, Nanjing 210023, Jiangsu, China
{212243039, 212243024, 212243029}@stu.njnu.edu.cn,
{lijuncst, xujianhua}@njnu.edu.cn

Abstract. Multi-label classification studies a problem where each instance is associated with multiple relevant labels, which leads to the exponential growth of output space. To address this issue of high-dimensional label space, dimensionality reduction strategy originally applied to feature space is also used in label space, known as label space dimensionality reduction (LSDR). One popular strategy to implement LSDR is label embedding (LE), which encodes the original high-dimensional label vector into a low-dimensional vector linearly or non-linearly. In this paper, We investigate the normalized cross-covariance operator (NOCCO), which originally is a kernel-based measure of the dependency between features and labels, whose empirical estimator is described as a trace operation including two inverse matrices of feature and label kernels plus a predefined regularization constant. We specifically designed an approximate and symmetric form of this operator for linear LE, which is maximized under orthonormal constraints, resulting in a novel eigenvalue problem for linear LE. The solution to this eigenvalue problem produces our compression matrix, and its transpose as our recovery matrix. Our proposed novel linear LE method based on maximizing normalized cross-covariance operator is termed as LEMCCO for short. The experiments on four benchmark data sets with more than 100 labels demonstrate that our proposed method is statistically superior to four state-of-the-art LE methods on the basis of two performance evaluation metrics.

Keywords: Multi-label classification · Dimensionality reduction · Label embedding

1 Introduction

Multi-label classification (MLC), which studies a problem of associating instances with multiple labels simultaneously [1], has garnered substantial attention in recent years due to its wide-ranging applications. However, MLC suffers

Supported by the Natural Science Foundation of China (NSFC) under grants 62076134 and 62173186.

from some challenges [11]. Apart from traditional large instance size and high-dimensional feature space, the size of label sets grows exponentially with the number of class labels [10], which results in a high computational burden and even an unsatisfactory classification performance.

To deal with this situation, label space dimensionality reduction (LSDR) is investigated, in which label embedding (LE) is a kind of representative techniques. The LE transforms the original label space into an lower-dimensional embedded space linearly or nonlinearly. In the meantime, it takes full advantage of the correlation among all labels, identifying the hidden structure of the original space. Correspondingly, many methods have been proposed. For example, PLST [12] essentially conducts principal component analysis on label space. It mainly obtains the projection matrix and decoder by performing the SVD on the label matrix efficiently. CPLST [2] introduces the feature space information into PLST. FaIE [9] aims to implicitly encode latent space to reduce the risk of inappropriate use of encoding functions. ML-mLV [13] constructs a trace ratio minimization problem as a novel label embedding criterion, which not only includes the global label recoverability and dependency, but also exploits the local label correlations as a local recoverability factor. LEDM [8] uses HSIC [5] to increase the correlation between the feature space and the label space, and it can be spread to track missing labels. ML-CCM [7] simplifies conditional covariance operator with linear label kernel matrix, which is then maximized under orthonormal constraints, resulting in an eigenvector problem.

Normalized cross-covariance operator (NOCCO) [4] is a kernel-based dependence measure between features and labels, whose empirical estimator is described as a trace operation including two inverse matrices of feature and label kernels plus a predefined regularization constant. This measure has been applied to object matching [16], feature extraction [15], feature selection [14] and independence test [3]. In our work, we explore the application of NOCCO for LE. However, due to its complex form, this measure cannot be directly converted into an eigenvalue problem. So we design an approximate and symmetric form of NOCCO specifically. Maximizing this new form generates a novel eigenvalue problem for linear LE under the orthonormal projection constraint. This novel label embedding method is referred to as the Label Embedding Algorithm based on Maximizing Normalized Cross-Covariance Operator (LEMCCO). Experiments conducted on four public datasets illustrate the effectiveness of our proposed label embedding approach, outperforming four state-of-the-art label embedding techniques according to two metrics.

In summary, our paper makes three main contributions: 1) designing an approximate and symmetric form of NOCCO for LE; 2) maximizing this new form to generate a novel eigenvalue problem for linear LE; and 3) conducting some extensive experiments to verify the effectiveness of our method.

This paper is organized as follows. In Sect. 2, we introduce some preliminaries. Then, our method is proposed in Sect. 3, which is validated in Sect. 4. Finally, we end up this paper with some conclusions.

2 Preliminaries

Let a multi-label training set of size N be

$$\{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_i, \mathbf{y}_i), \dots, (\mathbf{x}_N, \mathbf{y}_N)\}. \quad (1)$$

For the i -th instance, its feature vector is $\mathbf{x}_i = [x_{i1}, \dots, x_{iD}]^T \in R^D$, and its binary label vector is $\mathbf{y}_i = [y_{i1}, \dots, y_{iK}]^T \in R^K$. Here, the j -th entry of \mathbf{y}_i is set as 1 if the instance is associated with the j -th label and 0 otherwise. For convenience of formula representation, we also define the feature and label matrices \mathbf{X} and \mathbf{Y} :

$$\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]^T \in R^{N \times D}; \mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_N]^T \in \{0, 1\}^{N \times K}. \quad (2)$$

The traditional MLC is to learn a classifier $\mathbf{y} = f(\mathbf{x}) : R^D \rightarrow \{0, 1\}^K$, according to the above training set (1), which then is used to predict the binary label vectors for unseen instances. When a proper LE method is applied to MLC, we need to find out a label encoding operator ϕ and its decoding operator ϕ^{-1} . Via ϕ , each original label binary vector $\mathbf{y} \in \{0, 1\}^K$ is transformed into a k -dimensional real label vector $\mathbf{z} \in R^k$ or binary one $\mathbf{z} \in \{0, 1\}^k$, where $k < K$. Then a proper classifier ($\mathbf{z} \in \{0, 1\}^k$) or regressor ($\mathbf{z} \in R^k$) $\mathbf{z} = g(\mathbf{x})$ is trained, which is used to estimate a predicted label $\hat{\mathbf{z}}$ for a testing instance \mathbf{x} . According to ϕ^{-1} , we recover $\hat{\mathbf{z}}$ back to an K -dimensional binary predicted label vector $\hat{\mathbf{y}}$.

3 The Proposed Method

The empirical estimator for NOCCO is formulated as the following trace form [4]:

$$\text{NOCCO} = \text{trace}(\tilde{\mathbf{G}}_x \tilde{\mathbf{G}}_y) \quad (3)$$

where $\tilde{\mathbf{G}}_x$ and $\tilde{\mathbf{G}}_y$ are two new proxy matrices for feature and label data as follows:

$$\begin{aligned} \tilde{\mathbf{G}}_x &= \bar{\mathbf{G}}_x (\bar{\mathbf{G}}_x + \varepsilon \mathbf{I})^{-1} \\ \tilde{\mathbf{G}}_y &= \bar{\mathbf{G}}_y (\bar{\mathbf{G}}_y + \varepsilon \mathbf{I})^{-1} \end{aligned} \quad (4)$$

where $\bar{\mathbf{G}}_x = \mathbf{H} \mathbf{G}_x \mathbf{H}$ and $\bar{\mathbf{G}}_y = \mathbf{H} \mathbf{G}_y \mathbf{H}$. Here, \mathbf{G}_x and \mathbf{G}_y are two kernel matrices for feature and label spaces, respectively. Additionally, $\mathbf{H} = \mathbf{I} - \mathbf{u} \mathbf{u}^T / N$ is the centering matrix, where \mathbf{I} is the identity matrix and \mathbf{u} is the column vector with all one elements. Since there exists an inverse matrix with a predefined regularization constant ε in (4), the empirical NOCCO (3) could not be directly used to formulate an eigenvalue problem for LE task. Therefore, we adopted certain strategies to obtain its approximate form.

Firstly, when the original inverse matrix is replaced by the Moore-Penrose inverse matrix $\bar{\mathbf{G}}_x^+$ and the linear kernel $\mathbf{X} \mathbf{X}^T$ is applied to \mathbf{G}_x [15], The Eq. (4) can be transformed into:

$$\tilde{\mathbf{G}}_x^m = \mathbf{H} \mathbf{X} \mathbf{X}^T \mathbf{H} (\mathbf{H} \mathbf{X} \mathbf{X}^T \mathbf{H})^+. \quad (5)$$

Algorithm 1: Pseudo-code of our LEMCCO

Training Stage:**Input:**

\mathbf{X} and \mathbf{Y} : feature and label matrices.

Process:

Centralize feature and label matrices to obtain: $\bar{\mathbf{X}}$ and $\bar{\mathbf{Y}}$.

Calculate their Moore-Penrose inverse matrices: $\bar{\mathbf{X}}^+$ and $\bar{\mathbf{Y}}^+$.

Construct the matrix \mathbf{S} and the constrained maximization problem.

Solve the eigenvalue problem.

Build the projection matrix \mathbf{P} with the largest k eigenvalues.

Calculate reduced label matrix: $\mathbf{Z} = \bar{\mathbf{Y}}\mathbf{P}$.

Learn a regressor: $\mathbf{z} = g(\mathbf{x})$.

Output:

\mathbf{P} : projection matrix of size $K \times k$.

$g(\mathbf{x})$: a trained linear regressor.

Testing Stage:**Input:**

\mathbf{x} : a testing instance vector.

Process:

Calculate the k -dimensional real label vector $\mathbf{z} = g(\mathbf{x})$.

Detect a binary label vector \mathbf{y} using round operation: $\text{round}(\mathbf{P}^T \mathbf{z})$.

Output:

\mathbf{y} : a predicted high-dimensional binary label vector.

Let $\bar{\mathbf{X}} = \mathbf{H}\mathbf{X}$, therefore the above equation can be simplified to:

$$\tilde{\mathbf{G}}_x^a = \bar{\mathbf{X}}\bar{\mathbf{X}}^T(\bar{\mathbf{X}}\bar{\mathbf{X}}^T)^+ = \bar{\mathbf{X}}\bar{\mathbf{X}}^T(\bar{\mathbf{X}}^+)^T\bar{\mathbf{X}}^+ = \bar{\mathbf{X}}\bar{\mathbf{X}}^+. \quad (6)$$

Similarly, we substitute the linear kernel $\mathbf{Y}\mathbf{Y}^T$ for \mathbf{G}_y , the approximation form of the Eq. (4) is:

$$\tilde{\mathbf{G}}_y^a = \bar{\mathbf{Y}}\bar{\mathbf{Y}}^+ \quad (7)$$

Based on two Eqs. (6) and (7), we can derive the approximation form of NOCCO as:

$$\text{NOCCO}^a = \text{trace}(\bar{\mathbf{X}}\bar{\mathbf{X}}^+\bar{\mathbf{Y}}\bar{\mathbf{Y}}^+) = \text{trace}(\bar{\mathbf{Y}}^+\bar{\mathbf{X}}\bar{\mathbf{X}}^+\bar{\mathbf{Y}}) = \text{trace}(\mathbf{S}) \quad (8)$$

where $\mathbf{S} = \bar{\mathbf{Y}}^+\bar{\mathbf{X}}\bar{\mathbf{X}}^+\bar{\mathbf{Y}}$. It is important to note that this matrix \mathbf{S} is neither symmetric nor positive semidefinite. Therefore, this matrix \mathbf{S} may yield complex eigenvalues and eigenvectors, and thus may impede the sorting of eigenvalues. In this case, we use $(\mathbf{S} + \mathbf{S}^T)/2$ to replace \mathbf{S} to define an approximate and symmetric form of NOCCO.

$$\text{NOCCO}^{as} = \frac{1}{2}\text{trace}(\mathbf{S} + \mathbf{S}^T). \quad (9)$$

Table 1. The basic characteristics of the data sets

| Dataset | Instance | Features | Labels | Cardinality |
|-----------|----------|----------|--------|-------------|
| Bibtex | 7395 | 1836 | 159 | 2.40 |
| Bookmarks | 87856 | 2150 | 208 | 2.02 |
| Corel5k | 8000 | 499 | 374 | 3.52 |
| Corel16k | 6932 | 500 | 153 | 2.87 |

Assuming an orthogonal projection matrix $\mathbf{P} \in R^{K \times K}$, satisfying $\mathbf{P}^T \mathbf{P} = \mathbf{I}$, we can transform a label vector \mathbf{y} into another new label vector \mathbf{z} by multiplying it with \mathbf{P} , i.e., $\mathbf{Y} \rightarrow \mathbf{Y}\mathbf{P}$. By substituting $\mathbf{Y}\mathbf{P}$ back into the equation for \mathbf{S} , we can obtain an expression involving \mathbf{P} , \mathbf{X} , and \mathbf{Y} , which represents the relationship between the embedded labels and features

$$\mathbf{S} = (\overline{\mathbf{Y}}\mathbf{P})^+ \overline{\mathbf{X}}\overline{\mathbf{X}}^+ (\overline{\mathbf{Y}}\mathbf{P}) = \mathbf{P}^T [\overline{\mathbf{Y}}^+ \overline{\mathbf{X}}\overline{\mathbf{X}}^+ \overline{\mathbf{Y}}] \mathbf{P} = \mathbf{P}^T \mathbf{S}\mathbf{P} \quad (10)$$

where $\mathbf{P}^{-1} = \mathbf{P}^T = \mathbf{P}^+$. Substituting the new expression for \mathbf{S} into the Eq. (9), we will obtain a novel criterion describing the dependence between the feature space and the embedded labels.

$$\text{NOCCO}^{as} = \frac{1}{2} \text{trace}(\mathbf{P}^T \mathbf{S}\mathbf{P} + (\mathbf{P}^T \mathbf{S}\mathbf{P})^T) = \frac{1}{2} \text{trace}(\mathbf{P}^T (\mathbf{S} + \mathbf{S}^T) \mathbf{P}). \quad (11)$$

To address the label embedding task, we formulate the following constrained maximization problem:

$$\max \frac{1}{2} \text{trace}(\mathbf{P}^T (\mathbf{S} + \mathbf{S}^T) \mathbf{P}), \text{ s.t. } \mathbf{P}^T \mathbf{P} = \mathbf{I}. \quad (12)$$

This problem (12) could be converted into a standard eigenvalue problem via Lagrangian multiplier technique

$$\frac{1}{2} (\mathbf{S} + \mathbf{S}^T) \mathbf{P} = \Lambda \mathbf{P} \quad (13)$$

where the diagonal matrix Λ consists of K real eigenvalues in descending order. The time complexity to solve (13) is $O(K^3)$. The solution \mathbf{P} of this problem is composed of the k largest eigenvectors corresponding to the eigenvalues. The matrix \mathbf{P} directly forms the compression matrix, and the transpose of \mathbf{P} forms the recovery matrix. Finally, we summarize our new multilabel LE algorithm in Algorithm 1.

4 Experiments

To validate our proposed method, we conducted experiments on four benchmark databases using two multi-label classification evaluation metrics.

Table 2. The number of wins for each method and metric across four data sets

| Metric | PLST | CPLST | FaIE | ML-mLV | LEMCCO(Ours) |
|-------------|------|-------|------|--------|--------------|
| Precision@1 | 4 | 10 | 2 | 4 | 36 |
| Precision@3 | 4 | 12 | 3 | 6 | 30 |
| Precision@5 | 6 | 12 | 1 | 7 | 29 |
| DCG@3 | 4 | 11 | 2 | 7 | 31 |
| DCG@5 | 5 | 10 | 1 | 7 | 32 |
| Total wins | 23 | 55 | 9 | 31 | 158 |

4.1 Benchmark Data Sets and Evaluation Metrics

We choose four public-available multi-label datasets with more than 100 labels to assess the effectiveness of our method. The characteristics of the datasets are shown in detail in Table 1.

Two evaluation metrics for large-scale label sets [6] are adopted as our evaluation indexes: $Precision@n$ and (DisCounted Gain) $DCG@n$ ($n = 1, 2, 3$). For a testing instance \mathbf{x} , its ground label vector is $\mathbf{y} = [y_1, \dots, y_i, \dots, y_K]^T$ and predicted function values $\hat{\mathbf{y}} = [\hat{y}_1, \dots, \hat{y}_i, \dots, \hat{y}_K]^T$, these two metrics are defined as:

$$Precision@n = \frac{1}{n} \sum_{i \in rank_n(\hat{\mathbf{y}})} y_i; DCG@n = \frac{1}{n} \sum_{i \in rank_n(\hat{\mathbf{y}})} \frac{y_i}{\log_2(i+1)} \quad (14)$$

where $rank_n(\hat{\mathbf{y}})$ returns the top n label indexes of $\hat{\mathbf{y}}$. Finally, these two metrics would be averaged over all testing instances. Additionally, we will not consider $DCG@1$ since $DCG@1 = Precision@1$.

4.2 Compared Methods and Experimental Settings

In our experiments, we validate our method and four existing methods: PLST [12], CPLST [2], FaIE [9] and ML-mLV [13]. Here, the linear regressor is chosen as our baseline for all compared methods. Further, we divided the instances into a training set and a testing set for tenfold cross validation. In order to investigate how the reduced dimensionality affects the classification performance, we consider different reduced proportions from 10% to 100% of original label dimensionality K with a step 10% .

4.3 Performance Evaluation and Analysis

From Fig. 1, it can be observed that our algorithm has some disadvantages only in lower dimensions. In most other dimensions, it outperforms other algorithms. This is mainly due to our maximization of the dependency between feature space and reduced label space, as well as the algorithm’s capability to simultaneously generate the compression and recovery matrices.

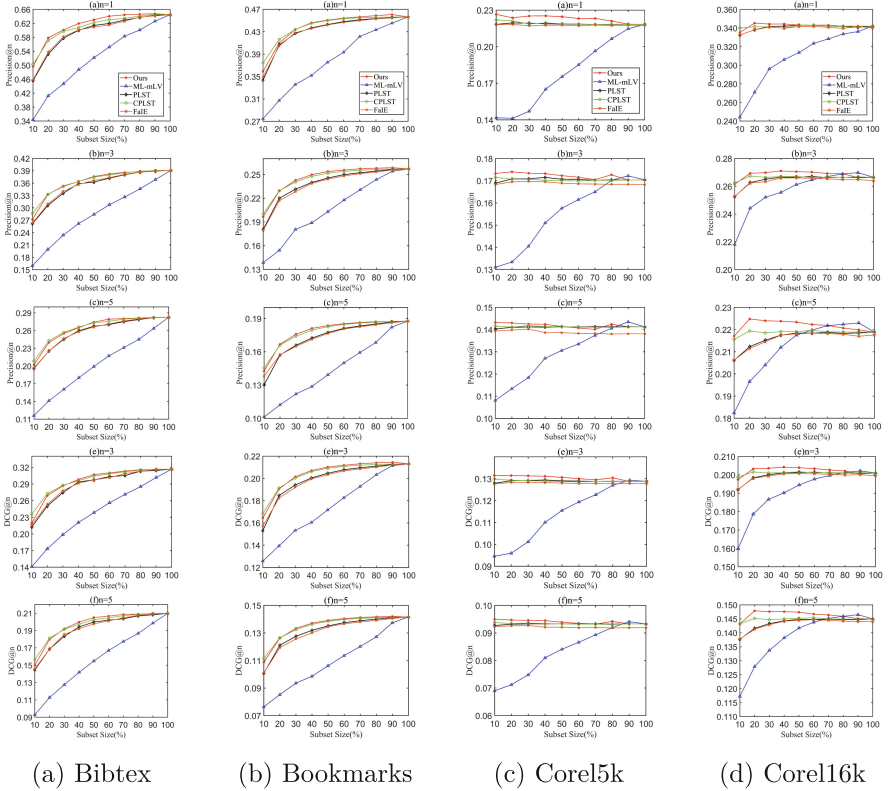


Fig. 1. Performance of the proposed LEMCCO using two metrics on the four benchmark data sets.

To provide a more intuitive comparison with these methods, we have also listed the number of times our method outperformed others in Table 2. It is found out that our LEMCCO achieve 158 wins among total 276 ones, which is greater than those of the other approaches.

5 Conclusion

In this paper, we focus on the application of NOCCO in the task of LE. We designed an approximate form of the original normalized cross-covariance operator that satisfies the orthogonal normalization constraint. This allowed us to construct a new linear label embedding algorithm capable of simultaneously generating dimensionality reduction and recovery matrices. Extensive experimental results demonstrate the superiority of our proposed method over four existing methods. In the future, we plan to validate the effectiveness of our approach on more benchmark datasets.

References

1. Charte, F., Rivera, A.J., Del Jesus, M.J.: *Multilabel Classification: Problem Analysis, Metrics and Techniques*. Springer, Switzerland (2016). <https://doi.org/10.1007/978-3-319-41111-8>
2. Chen, Y.N., Lin, H.T.: Feature-aware label space dimension reduction for multi-label classification. In: *25th Annual Conference on Neural Information Processing Systems*, pp. 1529–1537. MIT Press, Cambridge (2012)
3. Djonguet, T.K.M., Nkiet, G.M.: An independence test for functional variables based on kernel normalized cross-covariance operator. *J. Multivariate Anal.* **202**, 105293 (2024)
4. Fukumizu, K., Gretton, A., Sun, X., Scholkopf, B.: Kernel measures of conditional dependence. In: *20th Annual Conference on Neural Information Processing Systems*, pp. 489–496. MIT Press, Cambridge (2007)
5. Gretton, A., Bousquet, O., Smola, A., Scholkopf, B.: Measuring statistical dependence with hilbert-schmidt norms. In: Jain, S., Simon, H.U., Tomita, E. (eds.) *ALT 2005*, pp. 63–77. Springer, Heidelberg (2005). <https://doi.org/10.1007/11564089-7>
6. Jain, H., Prabhu, Y., Varma, M.: Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In: *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 935–944. ACM Press, New York (2016)
7. Li, D., Li, Y., Li, J., Xu, J.: A label embedding method via conditional covariance maximization for multi-label classification. In: Strauss, C., Amagasa, T., Kotsis, G., Tjoa, A.M., Khalil, I. (eds.) *DEXA 2023*, pp. 393–407. Springer, Heidelberg (2023). https://doi.org/10.1007/978-3-031-39821-6_32
8. Li, Y., Yang, Y.: Label embedding for multi-label classification via dependence maximization. *Neural Process. Lett.* **52**, 1651–1674 (2020)
9. Lin, Z., Ding, G., Hu, M., Wang, J.: Multi-label classification via feature-aware implicit label space encoding. In: *31st International Conference on Machine Learning*, pp. 325–333. MLResearch Press, Webside (2014)
10. Liu, W., Wang, H., Shen, X., Tsang, I.W.: The emerging trends of multi-label learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **44**(11), 7955–7974 (2022)
11. Peng, T., Li, J., Xu, J.: Label selection algorithm based on iteration column subset selection for multi-label classification. In: Strauss, C., Cuzzocrea, A., Kotsis, G., Tjoa, A.M., Khalil, I. (eds.) *DEXA 2022*, pp. 287–301. Springer, Heidelberg (2022). https://doi.org/10.1007/978-3-031-12423-5_22
12. Tai, F., Lin, H.T.: Multilabel classification with principal label space transformation. *Neural Comput.* **24**(9), 2508–2542 (2012)
13. Wang, X., Li, J., Xu, J.: A label embedding method for multi-label classification via exploiting local label correlations. In: Gedeon, T., Wong, K., Lee, M. (eds.) *ICONIP 2019*, pp. 168–180. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-030-36802-9_19
14. Xu, J., Lu, W., Li, J., Yuan, H.: Dependency maximization forward feature selection algorithms based on normalized cross-covariance operator and its approximated form for high-dimensional data. *Inf. Sci.* **617**, 416–434 (2022)
15. Xu, J., Mao, Z.H.: Multilabel feature extraction algorithm via maximizing approximated and symmetrized normalized cross-covariance operator. *IEEE Trans. Cybern.* **51**(7), 3510–3523 (2021)
16. Yamada, M., Sugiyama, M.: Cross-domain object matching with model selection. In: *14th International Conference on Artificial Intelligence and Statistics*, pp. 807–815. MLResearch Press, Webside (2011)



Analyzing the Efficacy of Large Language Models: A Comparative Study

Sonia Khetarpaul^(✉) , Dolly Sharma , Shreya Sinha , Aryan Nagpal ,
and Aarush Narang 

Department of Computer Science and Engineering, Shiv Nadar Institution
of Eminence, Delhi NCR, Greater Noida, India

{sonia.khetarpaul,dolly.sharma,ss115,an125,an229}@snu.edu.in

Abstract. With the rise of large language models (LLMs) and natural language processing (NLP) methods in businesses and industries, our research evaluates two prominent LLMs: GPT-3.5 Turbo by OpenAI and Llama 2 by Meta. We used an automated process to generate and fine-tune question-answer pairs, enhancing accuracy. Using established metrics, we quantified performance and developed a comprehensive evaluation metric. Our analysis highlighted the need for further improvements to address prevalent issues such as inaccuracies and ambiguities.

1 Introduction

LLMs like GPT-3.5 [3] by OpenAI and BERT [4] by Google, which represent advancements in machine learning (ML), utilize deep neural networks to generate human-like language. These models are widely used in text production, translation, and question answering. Consequently, companies leverage LLMs to build chatbots, analyze large textual datasets, and create personalized user experiences due to their impressive language capabilities. The increasing prevalence of LLMs necessitates a thorough examination of their strengths and weaknesses [2]. This study addresses accuracy concerns of LLMs by evaluating and comparing the performance of GPT-3.5 Turbo and Llama 2. We develop a comprehensive evaluation framework, beginning with generating and fine-tuning question-answer pairs, and use it to assess and compare these models.

2 Literature Review

LLMs are evaluated using various metrics to measure fluency, coherence, and relevance in generated text. ROUGE [6] and BLEU [8] scores are commonly used for this purpose. Additionally, METEOR [5] and BERT score [4] are notable metrics for assessing linguistic elements and contextual similarity, respectively. Perplexity score [7] evaluates the effectiveness of LLMs based on predictive accuracy, while cosine similarity [10] measures the similarity between vectors, frequently used in NLP tasks. The preliminary evaluation metrics for our framework include

the ROUGE score, which measures the similarity between generated and reference text; the BLEU score, which evaluates the similarity of translations; and cosine similarity, which measures the similarity between vectors. These metrics ensure a comprehensive assessment of LLM performance.

3 Datasets and Preprocessing

Before evaluating the LLMs’ performance, two crucial tasks were completed. First, we created a dataset of question-answer pairs for later evaluation. Second, the LLMs were trained and fine-tuned on our custom data to enable contextualized question answering.

3.1 Dataset Construction by Question-Answer Pair Generation

A dataset of 885 questions was curated, split between Cricket World Cup 2023 [13] and Israel-Hamas War data from Wikipedia [14]. The “Cricket World Cup ’23” dataset contains 280 true/false questions and 280 one-line answers. The “Israel-Hamas War” dataset includes 160 true/false questions and 165 one-line answers. In total, there are 440 true/false questions and 445 one-line answers. Each question was paraphrased using the paraphraser API for testing purposes [15]. The sample question-answer pairs generated using ChatGPT are shown in Table 1. The data extracted from Wikipedia was processed into JSON format, demonstrating the model’s capacity for automated information extraction and question generation, which streamlines dataset creation.

Table 1. Sample Question and Answers generated

| Questions | Answers |
|---|-----------------|
| True/False: England played against India on 29/10/23 | True |
| When did the ongoing armed conflict between Palestinian militant groups and Israel begin? | 17 October 2023 |

3.2 Fine-Tuning the LLM

Llama 2 is a collection of generative text models available in pretrained and fine-tuned versions, with parameter sizes ranging from 7B (Billion) to 70B. This research utilizes the 70B model of Llama 2, converted for Hugging Face Transformers. GPT-3.5 Turbo, boasting 16,385 tokens, offers advanced language comprehension and generation capabilities. Algorithm 1 outlines the complete procedure for answer generation.

Algorithm 1. Answer Generation**Require:** PDF raw data**Ensure:** Answer

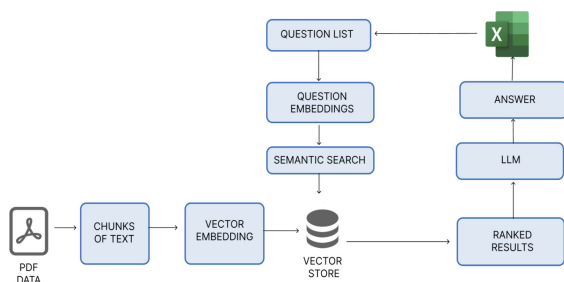
- 1: Split data into chunks and convert to embeddings.
- 2: Store embeddings in FAISS vector store.
- 3: **for** each question **do**
- 4: Convert question to embeddings and perform semantic search.
- 5: Pass top result to LLM generator to generate and store answer.
- 6: **end for**
- 7: Export answers to an Excel sheet.

3.3 Document Parsing and Text Comprehension

We transformed unstructured documents into a structured format by extracting text from diverse sources such as plain text and PDFs using Python and PyPDF. Text segmentation was achieved using LangChain Recursive TextSplitter, which segments text based on characters like periods or commas. In our framework, text comprehension involves storing knowledge within the model and retrieving it through Document-based Question-Answering (DocsQA). To facilitate this, text was converted into vector representations using OpenAI’s ‘text-embedding-ada-002’ model, stored in FAISS. Relevant text passages were identified, and answers were formulated within that context. Query and document fragments were embedded into vectors, and similarity searches were conducted to determine the most relevant chunks. Finally, the LLM generated answers based on conditional word probabilities [11].

4 Methodology

After completing the question-answer dataset and fine-tuning the LLM, we developed an evaluation framework similar to KMIR [1].

**Fig. 1.** Flowchart to process Q/A pairs

As shown in Fig. 1, we created reference questions and answers, then compared machine-generated responses to these using similarity scores. The dataset

was fed into the model, its responses recorded, and ROUGE [6], BLEU [8], and cosine similarity scores applied. A weighted sum of these scores was calculated to detect outliers, indicating incorrect answers by the LLM. Detailed steps for score calculation are provided in Algorithm 2.

Algorithm 2. Score Calculation

Require: LLM answers list, Human answer list

Ensure: Classification (True/False)

- 1: **for** each answer **do**
 - 2: Calculate BLEU, ROUGE, and cosine similarity scores.
 - 3: Compute final score = $0.1 \times \text{BLEU} + 0.4 \times \text{ROUGE} + 0.5 \times \text{Cosine similarity}$
 - 4: Push final score to list.
 - 5: **end for**
 - 6: Apply z-score outlier detection and declare threshold.
 - 7: **for** each score in list **do**
 - 8: Classify as True or False and push to list.
 - 9: **end for**
-

4.1 Integrated Evaluation: BLEU, ROUGE, and Cosine Similarity

In our evaluation framework, we combine the BLEU score, ROUGE score, and cosine similarity score into a weighted sum to reflect their relative importance and provide a comprehensive total score for the data points.

Balancing Metric Trade-offs: The proposed evaluation weights of 0.1 for BLEU, 0.4 for ROUGE, and 0.5 for cosine similarity are reasonable for evaluating our models. They emphasize semantic accuracy and content coverage, balancing meaning and informativeness with syntactic correctness.

4.2 Accuracy Assessment Through Z-Score Outlier Detection

Once we have a total weighted score for each question, we employ Z-Score outlier detection to identify answers with very low scores, indicating inaccuracies. This statistical method pinpoints outliers, particularly in normally distributed datasets, through a series of steps. We calculate the mean and standard deviation of the dataset, determine the Z-scores, and classify points beyond a specific threshold as outliers. Outliers are labeled as true for correct answers and false for incorrect ones.

5 Observations and Results

For the evaluation of our LLM with 885 test cases, metrics such as BLEU score, cosine similarity, and ROUGE score were employed, supported by robust outlier

detection for enhanced reliability. The research identified errors such as hallucinations, under-specification, and false negatives. A Python script, utilizing the GPT 3.5 Turbo model, classified answers as correct or incorrect.

As depicted in Fig. 2, GPT 3.5 Turbo achieved a higher answer accuracy of 70.9% compared to Llama 2, which attained an accuracy of 67.9%. This indicates that GPT 3.5 Turbo exhibits strength in providing factually correct answers, demonstrating a difference of 3% between the two models.

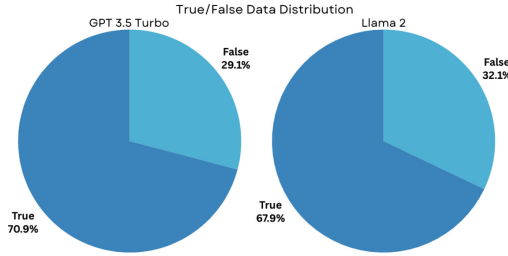


Fig. 2. Comparison of performance of LLMs

5.1 Classification of Errors

The errors were classified into the following categories:

1. **False Negative:** Correct answers marked as incorrect (8.5%).
2. **Hallucination:** Fabricated answers with no basis (40.6%).
3. **Under-specification:** Answers lacking key information (50.9%).

Table 2 shows sample test cases from our dataset. The type of data used to train LLMs affects the errors they make. For example, training on news articles might cause more errors due to biases and inaccuracies compared to training on more reliable sources [9].

Table 2. Sample Question and Answers generated

| Questions | Answers | LLM Answers | Correctness | Reason |
|---|---------------------------|--------------------------------|-------------|---------------|
| True/False: The tournament is being hosted by India | True | True | Correct | – |
| Which team secured their place as hosts? | India | India | Correct | – |
| What is the digital rights platform for broadcasting in India | Star Sports and DD Sports | Star Sports and Disney+Hotstar | Incorrect | Hallucination |

5.2 Comparison of Performance of Our Framework

Our framework was comprehensively evaluated across multiple parameters. Following result generation, we manually verified the model’s accuracy, categorizing each instance as correct or incorrect. For incorrect entries, we analyzed false positives and false negatives to assess performance.

Table 3. GPT 3.5-Turbo is better at providing correct answers

| Parameter | Model | |
|-----------------|--------|---------------|
| | LLaMA2 | GPT 3.5 Turbo |
| Answer Accuracy | 67.9% | 70.9% |
| Model Precision | 99.54% | 95.85% |
| Model Recall | 93.22% | 97.88% |
| Model F1 Score | 96.28% | 96.86% |
| Model Accuracy | 92.30% | 93.16% |

Table 3 compares our framework on Llama 2 and GPT 3.5 Turbo models, GPT-3.5 Turbo had a slightly higher accuracy rate (70.9% vs. 67.9%). Our framework’s performance on GPT-3.5 Turbo outperformed Llama 2 with higher accuracy, recall and a slightly better F1 score.

6 Conclusion and Future Directions

In this paper, we evaluated the performance of two prominent LLMs: GPT 3.5 Turbo by OpenAI and Llama by Meta. We utilized these models to generate and fine-tune Q/A pairs on datasets related to the Cricket World Cup and the Israel-Hamas War 2023. Metrics such as BLEU score, ROUGE score, and cosine similarity were calculated to evaluate model performance, and a combined metric was proposed to address score inconsistencies. Despite encountering common LLM challenges like hallucination and false negatives, GPT 3.5 Turbo achieved an accuracy of 70.9.

In our research, we recognize areas for future improvement in evaluation, including exploring a wider range of metrics, refining answers for accuracy, and involving human reviewers or experts for enhanced efficiency [12]. ChatGPT 3.5 Turbo was preferred over GPT 4 due to resource constraints and easier API accessibility, aiding smoother integration [16]. However, evaluating LLMs faces limitations such as incomplete representation by metrics like BLEU, ROUGE, and perplexity, as well as scalability issues due to computational resource requirements. Addressing these challenges is crucial for robust LLM evaluations meeting evolving language technology demands.

References

1. Gao, D., et al.: KMIR: a benchmark for evaluating knowledge memorization, identification and reasoning abilities of language models. arXiv preprint [arXiv:2202.13529](https://arxiv.org/abs/2202.13529) (2022)
2. Ayush, R., Pavan, R., Alisha, G.: Large Language Models (LLMs) – a backgrounder. Deloitte (2023)
3. OpenAI Platform. <https://platform.openai.com/docs/overview>
4. Devlin, J., et al.: BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL (2019). <https://aclanthology.org/N19-1423>
5. METEOR Score. Arize AI (2023). <https://arize.com/glossary/meteor-score>
6. Lin, C.-Y.: ROUGE: a package for automatic evaluation of summaries. Text summarization branches out (2004)
7. Chen, S.F., Beeferman, D., Rosenfeld, R.: Evaluation metrics for language models (1998)
8. Papineni, K., Roukos, S., Ward, T.J., Zhu, W.J.: BLEU (2001). <https://doi.org/10.3115/1073083.1073135>
9. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are few-shot learners. arXiv preprint [arXiv:2005.14165](https://arxiv.org/abs/2005.14165) (2019)
10. Rahutomo, F., Kitasuka, T., Aritsugi, M.: Semantic cosine similarity. In: The 7th International Student Conference on Advanced Science and Technology ICAST, vol. 4, no. 1 (2012)
11. effectz.AI. Creating Custom ChatGPT with Your Own Dataset using OpenAI GPT-3.5 Model, LLaMAIndex, and LangChain (2022)
12. Radford, A., et al.: Improving language understanding by generative pre-training. OpenAI **9** (2018)
13. 2023 Cricket World Cup. Wikipedia (2024). https://en.wikipedia.org/wiki/2023_Cricket_World_Cup
14. Isreal-Hamas war. Wikipedia (2024). https://en.wikipedia.org/wiki/Israel%E2%80%9393Hamas_war
15. Paraphrasing API Documentation. www.paraphraser.io/paraphrasing-api-documentation
16. Chang, Y.: A survey on evaluation of large language models. [arxiv.org:2307.03109](https://arxiv.org/abs/2307.03109) (2023)



Leveraging Large Language Models for Flexible and Robust Table-to-Text Generation

Ermelinda Oro^{1,2}(✉) , Luca De Grandis² , Francesco Maria Granata² ,
and Massimo Ruffolo^{1,2} 

¹ National Research Council, Institute for High Performance Computing
and Networking, via P. Bucci 8/9C, 87036 Rende, CS, Italy
`ermelinda.oro@cnr.it`

² Altilia srl, TechNest Start-up Incubator of University of Calabria, Piazza
Vermicelli, 87036 Rende, CS, Italy
`{luca.degrandis, francesco.granata, massimo.ruffolo}@altiliagroup.com`

Abstract. Generating natural language descriptions from structured tabular data is a crucial challenge with high-impact applications across diverse domains, including business intelligence, scientific communication, and data analytics. Traditional rule-based and machine learning approaches have faced limitations in reusability, vocabulary coverage, and handling complex table layouts. Recent advances in LLMs pre-trained on vast corpora offer an opportunity to overcome these limitations by leveraging their strong language understanding and generation capabilities in a flexible learning setup. In this paper, We conduct a comprehensive evaluation of two LLMs - GPT-3.5 and LLaMa2-7B - on table-to-text generation across three diverse public datasets: WebNLG, NumericNLG, and ToTTo. Our experiments investigate both zero-shot prompting techniques and finetuning using the parameter-efficient LoRA method. Results demonstrate GPT-3.5's impressive capabilities, outperforming LLaMa2 in zero-shot settings. However, finetuning LLaMa2 on a subset of data significantly bridges this performance gap and produces generations much closer to ground truth and comparable to SOTA approaches. Our findings highlight LLMs' promising potential for data-to-text while identifying key areas for future research.

Keywords: Natural Language Generation · Table-to-Text · Data-to-Text · LLM · Zero-Shot · GPT-3 · LLaMa · Prompt · Finetuning · LoRA

1 Introduction

The table-to-text task, which involves generating natural language descriptions from structured tabular data, is critical in diverse fields. It enables automatic generation in business, research finding summaries in science, and data

querying and summarization in analytics. This capability enhances understanding and accessibility of information across documents like spreadsheets, PDFs, and databases. Traditional rule-based and machine learning approaches for data-to-text generation have faced limitations in reusability, vocabulary coverage, and adaptability to diverse table layouts. In this paper, we propose leveraging the latest advances in large language models (LLMs) like GPT-3 and LLaMa to overcome these limitations. LLMs have demonstrated remarkable few-shot learning capabilities, attributed to their pre-training on vast text corpora, enabling flexible knowledge transfer to new tasks and domains. We hypothesize that with appropriate prompting and lightweight finetuning, LLMs can facilitate highly robust and generalizable table-to-text generation across diverse layouts and domains. Our main contributions are: (i) A comprehensive evaluation of two state-of-the-art (SOTA) LLMs - GPT-3.5 and LLaMa2-7B - on table-to-text tasks across three public datasets of varying complexity: WebNLG, NumericNLG, and ToTTo. (ii) Systematic exploration of both zero-shot prompting techniques and finetuning using the efficient Low-Rank Adaptation (LoRA) method. (iii) Demonstration of GPT-3.5’s impressive abilities, outperforming LLaMa2 in zero-shot settings, with finetuning significantly bridging LLaMa2’s performance gap. (iv) Achievement of competitive results by LoRA-tuned LLaMa2 compared to the SOTA, despite using fewer training data. The paper is structured as follows: Sect. 2 reviews related work, Sect. 3 describes methods and experimental setups, Sect. 4 presents results, and Sect. 5 concludes with insights and future directions.

2 Related Work

Researchers have explored various approaches to table-to-text generation, focusing on different table representations. A common approach is to preprocess tables to extract or highlight relevant information. For example, T5_{base-CoNT} [1] and Plan-then-Generate [11], utilize subtables from the ToTTo dataset, employing a process called ‘ToTTification’ to extract relevant table portions based on highlighted cells. Other approaches, such as TAsD [4] and T5_{template} [12], use template-based table serialization. Models like ExT5 [2], PaLM_{540B} [6], and T5_{CP} [7], are tuned on specific target datasets like WebNLG [8], which contain tables with simple structures and do not require complex table preprocessing. In our paper, we investigate LLMs’ capabilities in table-to-text tasks across datasets of varying complexity. We analyze their ability to handle full tables with rich contextual information, perform zero-shot learning, and adapt with limited training data. This work extends the understanding of LLMs’ versatility in data-to-text applications.

3 Methods and Experiment Settings

Task. Given a structured table containing categorical, numerical, or mixed data, the goal is to generate a natural language description that accurately and coherently conveys the key information in the table. The generated text should capture

salient facts, relationships, and insights from the tabular data, omitting irrelevant details, by understanding the table’s structure, content, and context to produce a fluent, informative summary.

Datasets. Three public table-to-text datasets with varying characteristics were used: (i) WebNLG [8] has triplet tables with concise whole-table summaries. (ii) NumericNLG [12] contains scientific numerical tables with descriptions referencing specified rows/columns and requiring logical inference. The remaining table content remains essential for logical inference. (iii) ToTTo [10] has extensive tables with summaries highlighting few selected cells. The datasets statistics are available in Table 1. A small random training subsample of 800 examples was used. Testing employed the complete WebNLG and NumericNLG test sets, and 10% (770 examples) of the ToTTo test set.

Table 1. Table-to-Text datasets.

| Dataset | Domain | #Documents (train/val/test) | Table size (#rows, #cols) | #words |
|-----------------|-------------------|-----------------------------|---------------------------|--------|
| WebNLG [3, 8] | General Purpose | 13,211/1,667/1,779 | 14.82, 3 | 19.77 |
| NumericNLG [12] | Scientific Tables | 1,084/136/135 | 8.13, 5.56 | 128.42 |
| ToTTo [10] | General Purpose | 120,761/7,700/7,700 | 32.87, 5.31 | 14.84 |

Metrics. To evaluate models’ text generation from tabular data, we use: (i) n-gram overlap metrics (BLEU, ROUGE, METEOR, TER, and PARENT) for syntactic assessment against reference descriptions and input tables. (ii) Entailment-based metrics (BERTScore, BLEURT) for semantic evaluation.

Models. Two models with contrasting dimensions and characteristics were utilized: Gpt-3.5-turbo-0613, a closed 175B parameter model accessed via the OpenAI API¹, and the open-source LLaMa2-7B model². LLaMa2-7B’s modest size enables single-GPU finetuning and deployment on smaller GPUs like Google Colab’s T4, providing a cost-effective text generation resource.

Table Formats and Prompts. During experiments, we prompted tables in three different formats: plain text, HTML, and JSON. This paper focuses on plain text for simplicity, as format was non-critical for the datasets considered, though complex/sparse tables warrant further investigation. For WebNLG triplets, headers labeled subject, relationship, and object were used. For NumericNLG, the input comprised tables and associated contexts, with prompts specifying target entities/locations. ToTTo had the largest tables, and performance may decline with size [5]. Therefore, three input formats were explored: (i) The full table with highlighted cells encapsulated within special tokens. (ii) A reduced version eliminating unnecessary rows/columns, (iii) A “ToTTified” format concatenating highlighted cells with row/column headers (SOTA preprocessing). Prompts for GPT-3.5 and LLaMa generations are reported in Table 2.

¹ <https://openai.com/>.

² <https://huggingface.co/meta-llama/Llama-2-7b>.

Table 2. Table-to-Text prompts

| Dataset | Prompt |
|-------------|---|
| Web-NLG | You are given a table in one of the following formats: html, plain text, or json.\nYour job is to use the table content to produce a short paragraph.\nThe paragraph must have the following properties:\n- it must be a few sentences long, if you believe that a single sentence is enough, you can use a single sentence\n- all the information in the table must be included in the paragraph\n- it must not include information that is not available in the table\n- it must not mention that the paragraph comes from a table\n\nTABLE:\n{table}\n\nPARAGRAPH:\n |
| Numeric-NLG | You are given a table in one of the following formats: html, plain text, or json.\nThe table is accompanied by a context and a target list. The targets are to be found in the row indexes.\nThe context can be in the form of table caption, title, or some other text.\nYour job is to use the table and the context to produce a paragraph.\nThe paragraph must have the following characteristics:\n- it must be structured in a way that allows the reader to understand it without seeing the table\n- it can't contain lists\n- it can only mention the entities in the target list\n\nTABLE:\n{plain_table}\n\nTARGET LIST\n{target_list}\n\nPARAGRAPH:\n |
| ToTTo | You are given a table in one of the following formats: html, plain text, or json.\nThe table is accompanied by a context.\nThe context is in the form of page title, table title, section title, and a few sentences taken from the same section as the table.\nThe table has some highlighted cells.\nYour job is to use the table and the context to produce a paragraph.\nThe paragraph must have the following characteristics:\n- it must be structured in a way that allows the reader to understand it without seeing the table\n- it can't contain lists\n- it must be only a single sentence long\n- it should focus on the highlighted cells preceded by <HIGHLIGHT_START> and followed by <HIGHLIGHT_END>\n- it must not explicitly mention that some cells are highlighted\n\nTABLE:\n{plain_table}\n\nCONTEXT:\n page title: {page.title}\n\nsection title: {section.title}\n\nsection text: {section.text} \n\n\nPARAGRAPH:\n |

Generation Techniques. Three main text generation approaches exist for large language models (LLMs): zero-shot (no examples), few-shot (prompts with few input-output examples), and fine-tuning (adjusting the pre-trained LLM’s parameters to a specific task or domain). We employed zero-shot prompting and fine-tuning. For fine-tuning, we utilized Parameter-Efficient Fine-Tuning (PEFT)³ [13], which modifies only a small subset of parameters. This mitigates the substantial computational resources typically required for fine-tuning, reducing training time and storage requirements, enabling LLM deployment on smaller machines without compromising performance. Specifically, we applied Low-Rank Adaptation (LoRA) [9] on a small random subset of 800 examples for training.

4 Experiments Results

This section presents the results of the automatic evaluation. Tables 3, 4, and 5 show the results on WebLG, NumericNLG, and ToTTo datasets respectively. In zero-shot experiments (the top portion of the tables), GPT outperformed LLaMa2 across syntactic and semantic metrics on all datasets, attributed to GPT’s larger model size. After finetuning (the bottom portion of the tables), the LoRA version of LLaMa2, trained on a subset, produced generations much

³ <https://github.com/huggingface/peft?tab=readme-ov-file>.

closer to ground truth and comparable to SOTA approaches. For ToTTo, table reduction yielded better results. Despite prompts instructing models to focus on highlighted cells, they still incorporated non-highlighted content, reducing scores. LLaMa2, finetuned with LoRA on 10% of training data using full tables, fell short of SOTA results. Whereas, finetuning on ToTTified tables significantly improved scores, nearly reaching SOTA levels despite using only 10% of data.

Table 3. Results on WebNLG.

| mModel | BLEU | R-1 | R-2 | R-L | METEOR | CHRf | TER | BERTScore | BLEURT |
|---------------------------|--------------|--------------|--------------|--------------|-----------|-------------|--------------|-------------|--------------|
| GPT-3.5 | 34.8 | 69.6 | 44.72 | 54.66 | 40.01 | 66.58 | 74.67 | 94.17 | 70.58 |
| LLaMa2 | 22.06 | 56.81 | 33.71 | 42.45 | 34.65 | 57.39 | 126.92 | 91.96 | 60.5 |
| ExT5 _{large} [2] | 35.03 | – | 48.17 | – | 36.5 | – | – | – | – |
| PaLM _{540B} [6] | 49.3 | – | – | – | – | – | – | – | – |
| T5 _{CP} [7] | 55.41 | – | – | – | 42 | 69.8 | 39.1 | – | 63.0 |
| LLaMa2 _{FT} | 51.42 | 78.14 | 53.91 | 62.92 | 40.72 | 69.43 | 43.46 | 95.4 | 75.55 |

Table 4. Results on NumericNLG.

| Model | BLEU | R-1 | R-2 | R-L | METEOR | CHRf | TER | PARENT | BERTScore | BLEURT |
|----------------------------|------|-------|-------|-------|--------|-------|--------|--------|-----------|--------|
| GPT-3.5 | 5.44 | 33.47 | 10.48 | 21.57 | 16.49 | 34.16 | 133.53 | 16.52 | 85.74 | 32.53 |
| LLaMa2 | 5.23 | 35.26 | 9.74 | 21.59 | 14.82 | 32.59 | 107.32 | 12.49 | 86.32 | 29.22 |
| T _{template} [12] | 5.02 | – | – | 30.25 | 20.11 | – | – | 15.09 | 87.68 | – |
| TASD [4] | – | – | – | 20.4 | 11.87 | – | – | – | – | – |
| LLaMa2 _{FT} | 5.71 | 36.47 | 14.18 | 27.08 | 12.48 | 25.91 | 88.11 | 14.44 | 87.55 | 33.62 |

Table 5. Results on ToTTo. The input table represents the tabular format given in the prompt.

| Model | Input table | BLEU | R-1 | R-2 | R-L | METEOR | CHRf | TER | PARENT | BERTScore | BLEURT |
|----------------------------|-------------|------|-------|-------|-------|--------|-------|---------|--------|-----------|--------|
| GPT-3.5 | Full | 8.3 | 37.04 | 18.91 | 27.92 | 26.75 | 40.53 | 290.1 | 39.05 | 88.11 | -0.484 |
| LLaMa2 | Full | 3.6 | 27.12 | 11.24 | 20.68 | 17.53 | 29.75 | 358.13 | 24.11 | 76.93 | -0.769 |
| GPT-3.5 | Reduced | 16.3 | 54.45 | 30.00 | 41.07 | 33.48 | – | 143.697 | 47.38 | 85.58 | -0.269 |
| GPT-3.5 | ToTTified | 18.7 | 56.57 | 31.65 | 43.68 | 34.70 | – | 118.36 | 47.52 | 87.09 | -0.081 |
| LLaMa2 | ToTTified | 15.3 | 51.80 | 27.18 | 38.86 | 30.98 | – | 126.517 | 38.48 | 85.53 | -0.188 |
| LLaMa2 _{FT} | Full | 28.0 | 54.81 | 34.04 | 46.88 | 26.03 | – | 72.06 | 36.93 | 85.26 | -0.218 |
| T _{base-CoNT} [1] | ToTTified | 49.1 | – | – | – | – | – | – | 58.9 | – | 0.238 |
| Plan-then-Generate [11] | ToTTified | 49.2 | – | – | – | – | – | – | 58.7 | – | 0.249 |
| LLaMa2 _{FT} | ToTTified | 45.1 | 67.91 | 44.38 | 57.39 | 35.39 | – | 59.51 | 56.43 | 90.58 | 0.176 |

5 Conclusion

This work investigated leveraging LLMs for table-to-text generation across the WebNLG, NumericNLG, and ToTTo datasets. We explored zero-shot prompting and parameter-efficient fine-tuning approaches. In zero-shot settings, GPT-3.5

outperformed LLaMa2, while fine-tuning LLaMa2 on a subset achieved competitive SOTA results, demonstrating LLMs' ability to generalize in a sample-efficient manner. However, generating text from complex, large tables remains challenging. Key future research directions include: (i) Developing improved prompting and fine-tuning strategies to enhance factual consistency and mitigate hallucinations. (ii) Modeling explicit representations of table structure and layout to better handle hierarchies and complexity. (iii) Defining and applying controllable text generation techniques to obtain personalized summaries considering highlighted content. (iv) Integrating knowledge graphs and explainable AI techniques to enhance reasoning capabilities and provide more interpretable and trustworthy text generation from tabular data. By continuing advances along these promising research avenues, we can revolutionize how structured data is communicated through natural language.

Acknowledgments. This work was partly supported by project FAIR - Future AI Research (PE00000013), under the NRRP MUR program funded by the EU-NGEU.

References

1. An, C., Feng, J., Lv, K., Kong, L., Qiu, X., Huang, X.: CONT: contrastive neural text generation (2023)
2. Aribandi, V., et al.: Ext5: towards extreme multi-task scaling for transfer learning (2022)
3. Castro Ferreira, T., et al.: The 2020 bilingual, bi-directional WebNLG+ shared task: overview and evaluation results (WebNLG+ 2020). In: International Workshop on Natural Language Generation from the Semantic Web (WebNLG+), pp. 55–76. ACL (2020)
4. Chen, M., et al.: Towards table-to-text generation with pretrained language model: a table structure understanding and text deliberating approach (2023)
5. Chen, W.: Large language models are few (1)-shot table reasoners. In: Findings of the Association for Computational Linguistics: EACL, pp. 1120–1130. ACL (2023)
6. Chowdhery, A., et al.: Palm: scaling language modeling with pathways. *J. Mach. Learn. Res.* **24**(240), 1–113 (2023)
7. Clive, J., Cao, K., Rei, M.: Control prefixes for parameter-efficient text generation. In: Workshop on Natural Language Generation, Evaluation, and Metrics (GEM), pp. 363–382. ACL (2022)
8. Gardent, C., Shimorina, A., Narayan, S., Perez-Beltrachini, L.: Creating training corpora for NLG micro-planners. In: ACL, pp. 179–188. ACL (2017)
9. Hu, E.J., et al.: Lora: low-rank adaptation of large language models (2021)
10. Parikh, A., et al.: ToTTo: a controlled table-to-text generation dataset. In: EMNLP, pp. 1173–1186. ACL (2020)
11. Su, Y., Vandyke, D., Wang, S., Fang, Y., Collier, N.: Plan-then-generate: controlled data-to-text generation via planning (2021)
12. Suadaa, L.H., Kamigaito, H., Funakoshi, K., Okumura, M., Takamura, H.: Towards table-to-text generation with numerical reasoning. In: ACL and IJCNLP, pp. 1451–1465. ACL (2021)
13. Xu, L., Xie, H., Qin, S.Z.J., Tao, X., Wang, F.L.: Parameter-efficient fine-tuning methods for pretrained language models: a critical review and assessment (2023)

Recommender Systems and Personalization



Collaborative Filtering for the Imputation of Patient Reported Outcomes

Eric Ababio Anyimadu¹✉, Clifton David Fuller², Xinhua Zhang³,
G. Elisabeta Marai³, and Guadalupe Canahuate¹

¹ Electrical and Computer Engineering, University of Iowa, Iowa City, IA 52242, USA
eric-anyimadu@uiowa.edu

² Department of Radiation Oncology, The University of Texas,
MD. Anderson Cancer Center, Houston, TX, USA

³ Department of Computer Science, University of Illinois Chicago,
Chicago, IL 60607, USA

Abstract. This study addresses the prevalent issue of missing data in patient-reported outcome datasets, particularly focusing on head and neck cancer patient symptom ratings sourced from the MD Anderson Symptom Inventory. Given that many data mining and machine learning algorithms necessitate complete datasets, the accurate imputation of missing data as an initial step becomes crucial. In this study we propose, for the first time, the use of collaborative filtering for imputing missing head and neck cancer patient symptom ratings. Two configurations of collaborative filtering, namely patient-based and symptom-based, leverage known ratings to infer the missing ones. Additionally, this study compares the performance of collaborative filtering with alternative imputation methods such as Multiple Imputation by Chained Equations, Nearest Neighbor Imputation, and Linear interpolation. Performance is compared using Root Mean Squared Error and Mean Absolute Error metrics. Findings demonstrate that collaborative filtering is a viable and comparatively superior approach for imputing missing patient symptom data.

Keywords: Head and Neck Cancer · Imputation · Collaborative Filtering

1 Introduction

Head and neck cancer (HNC) patients often experience disease-related symptoms and side effects during and after treatment which can affect their quality of life and survival [16]. Researchers and physicians are therefore increasingly placing significant attention on leveraging existing patient symptom data to personalize care for patients and improve patient outcomes [17]. Furthermore, the examination of patient symptom data has been recognized as having the capacity to yield fresh insights into clinical understanding to enhance diagnostic accuracy and optimize the effective allocation of healthcare resources [15].

The MD. Anderson Symptom Inventory (MDASI) is a validated instrument to collect patient reported outcomes. The MDASI Head and Neck module (MDASI-HN) is a 28-symptom questionnaire relevant for head and neck cancer patients [20]. Patient responses are collected before, during, and after treatment and similar to other longitudinal datasets that rely on patient responses or feedback, the MDASI-HN data often contain missing values [1, 24]. This imposes restrictions on the applicability of numerous statistical methods and machine learning approaches in analyzing these incomplete datasets, given that these techniques usually require complete datasets [5]. Moreover, discarding data from patients with missing responses in order to achieve complete datasets may introduce bias in parameter estimation. These patients could possess special characteristics that are not representative of the broader group, thus limiting the extent to which these analyses can be generalized [2, 25]. To address this issue, missing values are commonly imputed as an initial step.

Several techniques exist for imputation, including Multiple Imputation by Chained Equations (MICE), K Nearest Neighbor (KNN) methods, and Linear Interpolation (LI) [7, 11, 24]. Despite their effectiveness in various scenarios, these methods may fail to capture intricate data relationships, particularly regarding patient sensitivity which are influenced by individual tolerance levels.

Collaborative filtering, a technique successfully employed in recommendation systems to leverage user preferences for personalized suggestions, offers a promising alternative [10]. We hence propose and evaluate the use of collaborative filtering to impute missing responses in the MDASI-HN, leveraging similarities in reported outcomes to enhance imputation accuracy.

Furthermore, we conduct experimental analyses comparing the performance of collaborative filtering against other established methodologies. Performance metrics used for evaluation include root mean square error and mean absolute error.

2 Related Work

We reviewed related work in two main categories: studies on the imputation of HNC symptom data and studies on collaborative filtering and its applications.

Imputation utilizes existing data and inherent associations to forecast specific or range-based approximations for missing values. Over the past few years, some studies have employed one imputation technique or the other in filling missing values in HNC symptom data. Some of the widely used imputation techniques are MICE, KNN and LI. MICE iteratively fills missing values in a dataset, creating a complete set of data in each cycle, improving with each iteration until an ultimate dataset is achieved [11, 13]. Conversely, KNN leverages intrinsic patient similarities to infer missing outcomes, while LI estimates values assuming a linear relationship [24].

Relevant studies in this field include one focused on the impact of radiation-induced toxicities on the quality of life for patients treated for HNC, which utilized MICE to complete both physician-rated toxicities and patient-rated symptoms post-radiotherapy [11]. Another study on using a Long Short-Term Memory

(LSTM) neural network to predict late-stage symptom severity demonstrated the effectiveness of imputation techniques, including LI and MICE, for addressing missing data in MDASI-HN patient-reported outcomes [24]. Additionally, a study on predicting clinical outcomes of radiotherapy in HNC patients employed statistical, MICE, and KNN imputation methods, highlighting the superior performance of MICE compared to the other techniques [7].

While these and other studies have employed various techniques to fill missing values, they primarily used these methods as pre-processing steps without focusing on comprehensive evaluations of the imputation techniques.

Collaborative filtering (CF) methods are widely used in recommendation systems such as GroupLens, Amazon.com, Netflix, Google News, and Facebook and excel in predicting user preferences based on collected ratings [19]. CF methods have been proposed for data imputation as well. The auto-adaptive CF imputation method, which leverages both item and user ratings to predict missing values and validated using the MovieLens dataset, was shown to outperform traditional imputation techniques [14]. Similarly, CF method based on rough-set theory was applied for imputing missing values in microarray gene expression data [23]. This CF based method outperformed KNN method over changing rates of missing values.

These studies showed the viability of CF in imputing missing values in a wide variety of fields. Nonetheless, to the best of our knowledge, CF has not been applied for the imputation of MDASI-HN patient-reported outcomes data before. Our objective is to demonstrate the effectiveness of collaborative filtering compared to established methods in this context. We employ traditional CF methods as a foundation, paving the way for future research on this topic using MDASI-HN data.

3 Methodology

In this section, we begin by describing the data. Next, we introduce the collaborative filtering technique and explain how we used it to fill missing patient-reported outcomes.

3.1 MDASI-HN Data

The MDASI-HN 28 questionnaire items are categorized as follows: 13 core MDASI items that rate general cancer symptoms, 9 HNC-specific items that rate symptoms associated with HNC and 6 interference items that assess how severely symptoms interfere with daily activities [20].

Each patient self-reports the 28 symptoms on a 0–10 scale with 0 indicating “not present” and 10 indicating “as bad as you can imagine”. Patients are asked to rate each item according to its worst severity during the previous 24 h [21].

All HNC patients in the cohort underwent standard of care treatment (radiotherapy with or without chemotherapy) with curative intent. The HNC patients completed the MDASI-HN questionnaires at the following stages: baseline ratings before the start of treatment, weekly evaluations spanning 7 weeks throughout the treatment course, and additional assessments after the 6th week as well as at the 6th, 12th, and 18th months after completion of treatment. The MDASI questionnaires can be abstracted as a two-dimensional user-item matrix where rows correspond to patients and columns correspond to symptoms.

We denote as $R_{p,i}$ the rating for patient p and symptom i . We distinguish between different time points, denoting as $R_{p,i}^t$, the rating provided by patient p for symptom i at time point t . For patients with missing ratings, $R_{p,i}^t = NA$.

In addition to symptom data, the dataset also includes clinical information of each patient such as biographic information (age, sex, change in height and weight during treatment), disease specifics (site of tumor, new disease after primary and TNM stage) and treatment information (prior treatment at enrolment, induction or concurrent chemotherapy, neck dissection and surgery status).

3.2 Collaborative Filtering (CF) for MDASI-HN

CF methods leverage the similarity between known preferences of the users without requiring the use of other external information to predict unknown preferences [10,22]. There are two variations of the CF techniques: the user-based which leverages similarity between users and the item-based method which exploits the similarity between items [22].

Let's consider an example using the user-based CF approach for book recommendations. The users provide book ratings to indicate their book preferences (e.g. likes and dislikes). Given the current preferences of a user p and the preferences of all other users, we are seeking to predict whether user p would like book i . The first step is to identify users that have rated book i and select the top k users ranked by the similarity of their preferences to the preferences of p . The average rating from the k users is used to predict the rating user p would give to book i . For the item-based approach, all the existing ratings for book i are compared against the ratings for all other books user p has rated and the top k most similar ones are used to predict the rating for book i for user p .

We adapt the user-based and item-based CF approaches to derive the CF Patient-based and CF Symptom-based methods to predict missing symptom ratings as explained below.

CF Patient-Based (CF-PAT) approach predicts missing ratings using known ratings from other patients who are most similar to the given patient. The procedure to impute a missing rating for symptom i at time-point t by patient p represented as $R_{p,i}^t$ using CF-PAT imputation is as follows:

- Find all patients Q who have known ratings for symptom i at time-point t .
- Determine the similarity, $\text{sim}(p, q)$, between patient p and each $q \in Q$ using their common known ratings.
- Select the top k of these q , i.e. $q[1], \dots, q[k]$ patients that are most similar to p .
- Calculate the missing rating as the average of the ratings of symptom i at time-point t by k weighted by their similarity measure as shown in Eq. 1:

$$R_{p,i}^t = \frac{\sum_k R_{q[k],i}^t * \text{Sim}(p, q[k])}{\sum_k \text{Sim}(p, q[k])} \quad (1)$$

where $\text{Sim}(p, q)$ is a patient similarity derived using a similarity metric (see Table 1) over the common ratings between patients p and q .

CF Symptom-Based (CF-SYM) on the other hand predicts missing ratings using known ratings from other selected symptoms or time-points rated by the same patient. This selection is guided by the inter-symptom relationships identified across all patients. The process to impute missing rating for symptom i at time-point t by patient p denoted as $R_{p,i}^t$ is derived as follows:

- Find all symptoms J where ratings for patient p are known.
- Using a similarity metric, determine the similarity measure between symptom i and all $j \in J$ symptoms using their common known ratings among all patients.
- Select the top k of j , i.e. $j[1], \dots, j[k]$ that are most similar to i using their similarity measures.
- Impute the missing rating as the average ratings of the k symptoms rated by p weighted their similarity measure to symptom i as shown in Eq. 2:

$$R_{p,i}^t = \frac{\sum_k R_{p,j[k]}^* * \text{Sim}(i, j[k])}{\sum_k \text{Sim}(i, j[k])} \quad (2)$$

where $\text{Sim}(i, j)$ is a symptom similarity derived using a similarity metric (see Table 1) over the common ratings between symptoms i and j . The notation $R_{p,j}^*$ is used to indicate that each symptom time point is considered independently when computing the similarity between symptoms and the most similar time points are used for rating imputation.

In both the CF-PAT and CF-SYM configurations, the top 5 most similar neighboring patients or symptoms are selected (i.e. $k = 5$). The process described above is repeated until all missing ratings are filled.

Table 1. CF Similarity Metrics. For Patient-based similarity, N represents the count of shared ratings between patients p and q . The symbols \hat{R}_p and \hat{R}_q used in PCC denote the average ratings of the shared ratings between patients p and q , respectively. For Symptom-based similarity, N signifies the number of patients who have rated both symptoms i and j , and \hat{R}_i and \hat{R}_j represent the averages of the common ratings between symptoms i and j , respectively.

| | Patient-based | Symptom-based |
|---------------------------------|---|---|
| Euclidean similarity | CF-PAT-EUC | CF-SYM-EUC |
| | $\text{Sim}(p, q) = 1 - \sqrt{\sum_{i=1}^N (R_{p,i} - R_{q,i})^2}$ | $\text{Sim}(i, j) = 1 - \sqrt{\sum_{p=1}^N (R_{p,i} - R_{p,j})^2}$ |
| Cosine similarity | CF-PAT-COS | CF-SYM-COS |
| | $\text{Sim}(p, q) = \frac{\sum_{i=1}^N R_{p,i} \cdot R_{q,i}}{\sqrt{\sum_{i=1}^N (R_{p,i})^2 \cdot (R_{q,i})^2}}$ | $\text{Sim}(i, j) = \frac{\sum_{p=1}^N R_{p,i} \cdot R_{p,j}}{\sqrt{\sum_{p=1}^N (R_{p,i})^2 \cdot (R_{p,j})^2}}$ |
| Pearson correlation coefficient | CF-PAT-PCC | CF-SYM-PCC |
| | $\text{Sim}(p, q) = \frac{\sum_{i=1}^N (R_{p,i} - \hat{R}_p) \cdot (R_{q,i} - \hat{R}_q)}{\sqrt{\sum_{i=1}^N (R_{p,i} - \hat{R}_p)^2 \cdot (R_{q,i} - \hat{R}_q)^2}}$ | $\text{Sim}(i, j) = \frac{\sum_{p=1}^N (R_{p,i} - \hat{R}_i) \cdot (R_{p,j} - \hat{R}_j)}{\sqrt{\sum_{p=1}^N (R_{p,i} - \hat{R}_i)^2 \cdot (R_{p,j} - \hat{R}_j)^2}}$ |

Similarity Metrics: We experimented with three commonly used similarity metrics in both the CF-PAT and CF-SYM techniques [9]. These similarity metrics were Euclidean similarity (EUC), the vector-based Cosine similarity (COS) and the correlation-based Pearson Correlation Coefficient similarity (PCC) [9, 22].

EUC is a linear metric and has gained widespread applicability due to its simplicity and effectiveness, particularly in the analysis of non-sparse numerical data [9]. Meanwhile, COS approach treats sets of ratings as vectors, calculating the cosine angle between them. This method carries the advantage of naturally providing a normalized distance measure. PCC also measures the linear relationship between two sets of ratings, expressed as the ratio of their covariance to the standard deviation [9, 22].

Each of these similarity measures contributed uniquely to the analyses, catering to different aspects of similarity evaluation in the dataset. Table 1 provides the different equations used to determine the various measures of similarity using EUC, COS or PCC in the CF-PAT or CF-SYM configurations.

Note that to ensure consistency, all similarity values were normalized to range from 0, signifying no similarity, to 1, representing the highest degree of similarity.

4 Evaluation

To assess the performance of the imputation techniques, in addition to the original missing values, we randomly masked some known values to serve as our ground truth data per symptom. We assumed that patients provided at least one rating for each symptom throughout the monitoring period and hence during the masking process, we ensured that every patient retained at least one known rating for each symptom.

We evaluated both the patient-based (CF-PAT) and symptom-based (CF-SYM) versions of the collaborative filtering (CF) methods, employing the three distinct similarity measures: Euclidean distance (EUC), Cosine similarity (COS), and Pearson correlation coefficient (PCC). Furthermore, we explored more adaptations of CF-PAT, considering the different treatment stages (baseline, during treatment, and post-treatment) independently, which we termed Per Treatment Stage (PTS).

Also, a prerequisite for calculating the similarity was to have an arbitrary minimum of 10 common ratings between patients or symptoms to ensure reliability of the measurements.

We compared the performance of the nine CF-based methods against three established methods: MICE, KNN imputation, and LI.

We applied the MICE technique with two different configurations: MICE with clinical data (MICE-w-Clinical) and MICE with only ratings (MICE-Ratings). The KNN method computed similarity between patients using Euclidean similarity over the available clinical data. Additionally, the LI method filled missing values for each patient and symptom independently, leveraging known patient ratings for a given symptom at various time points.

4.1 Evaluation Metrics:

As is commonly used in evaluating machine learning models, we assessed imputation performance using root mean squared error (RMSE) and mean absolute error (MAE) measurements [8].

MAE is a linear error measurement, implying that all individual deviations are assigned equal importance in determining the overall result making it a more natural measure of average error. On the other hand, RMSE calculates the average magnitude of squared errors and consequently assigns relatively higher weight to larger errors making it comparatively more sensitivity [8].

RMSE and MAE over T imputation points are derived as shown in Eqs. 3 and 4 respectively:

$$RMSE = \sqrt{\frac{1}{T} \sum_{t=1}^T (P_t - A_t)^2} \quad (3)$$

$$MAE = \frac{1}{T} \sum_{t=1}^T \|P_t - A_t\| \quad (4)$$

where P and A are the sets of imputed and actual/ground truth data respectively. Smaller values of RMSE/MAE indicate a better performance.

5 Experimental Results

5.1 Experimental Setup

To inject missing values in the data, random masking was performed to remove an average of 2%-3% of the original values. All missing values were then imputed, and using the ground truth from the masked values, the imputation methods were evaluated using RMSE and MAE.

We repeated the experiments five times, each time randomly generating masked versions of the dataset and reported the average evaluation metric scores for each method.

We performed all the analyses using python 3.11 version. Python scikit learn, numpy and pandas were used for data pre-processing and implementation of the imputation techniques. The experiments were conducted on a MacBook Pro.

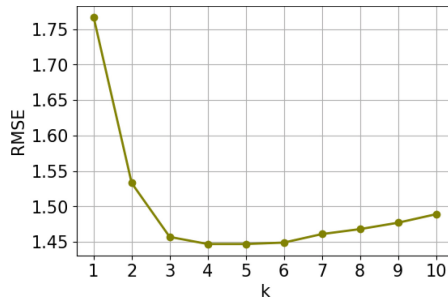


Fig. 1. RMSE of CF-SYM-PCC imputation over changing number of selected neighbors (k).

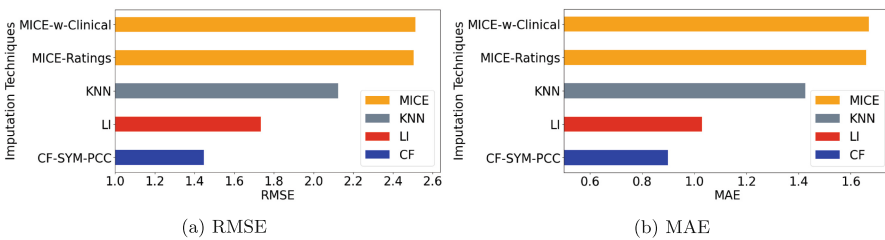


Fig. 2. Comparison of the imputation techniques. Over the masked or ground truth dataset, CF-SYM-PCC was the best imputation method using either (a) RMSE or (b) MAE values.

5.2 Data Statistics

The data for the analyses encompassed a cohort of 821 patients, and Table 2 provides a breakdown of the distribution and frequency of missing symptom

ratings among these patients, stratified based on their clinical data. To ensure consistency and eliminate discrepancies arising from diverse measurement scales all categorical clinical records were transformed into binary representations using one-hot encoding, while numerical values were normalized to a range of 0 to 1 before use in the analyses [18].

Table 2. Cohort Distribution and Missing Symptom Rate Stratified By Clinical Data.

| Features | Categories | Distribution of Patients (%) | Rate of Missing Ratings (%) |
|--------------------------------|----------------|------------------------------|-----------------------------|
| Biographical Data | | | |
| Age | <60 years | 47.25 | 20.50 |
| | ≥ 60 years | 52.75 | 23.00 |
| Sex | Female | 11.32 | 5.27 |
| | Male | 88.68 | 38.10 |
| Height change during treatment | Increase | 2.10 | 0.81 |
| | Decrease | 1.06 | 0.59 |
| | No change | 96.84 | 42.45 |
| Weight change during treatment | Increase | 4.50 | 1.94 |
| | Decrease | 32.28 | 11.95 |
| | No change | 63.22 | 29.94 |
| Disease Data | | | |
| Site of Tumor | Base of tongue | 44.62 | 18.06 |
| | Tonsil | 44.00 | 19.44 |
| | Others | 4.37 | 46.41 |
| | Not specified | 7.01 | 2.75 |
| New disease after primary | Yes | 5.11 | 1.06 |
| | No | 94.89 | 15.39 |
| T-stage | t0 | 7.08 | 2.56 |
| | t1 | 29.33 | 11.50 |
| | t2 | 37.42 | 15.55 |
| | t3 | 13.65 | 5.84 |
| | t4 | 11.88 | 5.63 |
| | tx | 0.64 | 0.26 |
| N-stage | n0 | 12.77 | 6.51 |
| | n1 | 34.77 | 14.31 |
| | n2,a,b,c | 49.94 | 40.43 |
| | n3,a | 2.15 | 50.95 |
| | nx | 0.37 | 0.06 |
| M-stage | 1 | 5.04 | 2.11 |
| | 2 | 94.96 | 32.79 |

(continued)

Table 2. (continued)

| Features | Categories | Distribution of Patients (%) | Rate of Missing Ratings (%) |
|-------------------------|----------------------|------------------------------|-----------------------------|
| Treatment Data | | | |
| Status at Enrollment | Previously Treated | 5.16 | 2.20 |
| | Previously Untreated | 94.84 | 41.28 |
| Induction Chemotherapy | Yes | 23.42 | 9.36 |
| | No | 76.58 | 26.78 |
| Concurrent Chemotherapy | Yes | 59.21 | 23.65 |
| | No | 40.79 | 17.90 |
| Neck Dissection | Yes | 75.27 | 9.46 |
| | No | 24.73 | 24.18 |
| Surgery at Primary Site | Yes | 80.22 | 8.28 |
| | No | 19.78 | 25.15 |

The rate of missing symptoms originally in the data and average rate over five random masks according to the treatment stages are as follows: baseline (original: 19.66%, after masking: 21.71%), during treatment (original: 50.57%, after masking: 53.05%) and post-treatment (original: 42.89%, after masking: 45.38%).

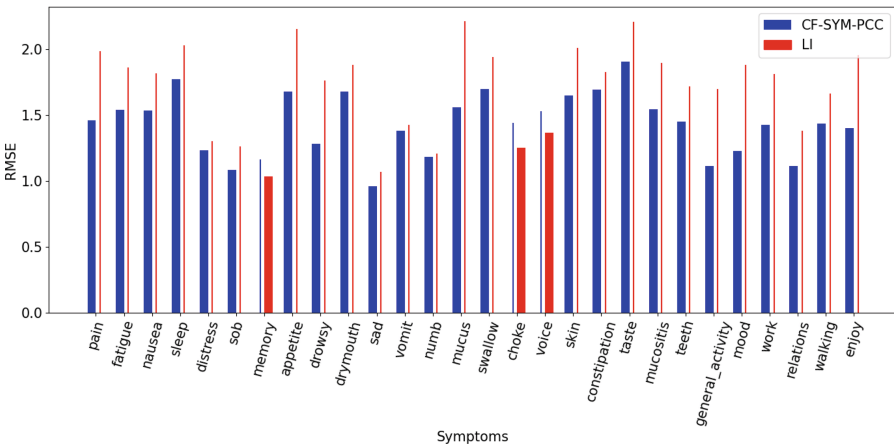


Fig. 3. RMSE comparison between CF-SYM-PCC and LI methods per symptom. The best (smaller) RMSE for each symptom is represented by a thicker bar.

5.3 CF Techniques Comparison

Table 3 shows the RMSE and MAE results for the evaluation of the nine variations of CF techniques, sorted by MAE from best to worse. In general, the CF-SYM techniques outperformed the CF-PAT methods. This superior performance of CF-SYM can be attributed to several factors. Firstly, the association between symptoms tends to be more established than that between patients as symptoms often exhibit clearer patterns of co-occurrence [6, 26]. This is further supported by research which indicates that symptom-based models can very effectively capture underlying disease and treatment responses [4]. Additionally, each symptom at a given time point has more ratings from individual patients compared to the number of ratings provided by each patient for the fewer number of symptoms. Consequently, CF-SYM leverages a larger number of ratings for determining similarity relative to CF-PAT, resulting in a more reliable and robust selection of high-quality neighbors or collaborators and hence better accuracy of the CF-SYM imputations.

In terms of the similarity metrics, PCC emerged as the optimal for CF-SYM. As compared to the other metrics, PCC in determining similarity normalizes all ratings by subtracting the mean of common ratings between each pair of symptoms. This mean normalization scales all ratings used in computing the PCC similarity uniformly therefore making the levels of ratings comparable regardless of the actual numeric values. Mean scaling therefore reduces the variations between ratings and ensures similar patterns in symptom ratings are identified for imputation. As a result, PCC is more effective at identifying nuanced correlations that might be overlooked by EUC and COS, leading to more precise and reliable similarity assessments in CF-SYM.

Overall, the best CF method for missing value imputation was CF-SYM-PCC under both RMSE and MAE metrics. Therefore, for the rest of this section, we focus on the performance of CF-SYM-PCC and its comparison with other methods.

Table 3. CF Techniques Comparison Results

| CF Techniques | RMSE | MAE |
|-------------------|--------------|--------------|
| CF-SYM-PCC | 1.447 | 0.898 |
| CF-SYM-EUC | 1.738 | 0.998 |
| CF-SYM-COS | 1.739 | 1.023 |
| CF-PAT-EUC-PTS | 1.737 | 1.048 |
| CF-PAT-EUC | 1.843 | 1.107 |
| CF-PAT-PCC | 1.797 | 1.127 |
| CF-PAT-PCC-PTS | 1.813 | 1.171 |
| CF-PAT-COS-PTS | 1.842 | 1.239 |
| CF-PAT-COS | 1.842 | 1.239 |

5.4 Effect of k on CF-SYM-PCC Imputation

Figure 1 illustrates the effect of varying the number of selected neighbors, represented by k , on the performance of the CF-SYM-PCC technique. Notably, the optimal performance is seen for k being 4 or 5 (RMSE: 1.447), while the least desirable performance was observed at $k = 1$ (RMSE: 1.767). The pattern follows the observed behavior of KNN approaches in other works. When k values are exceedingly small, collaboration effectiveness may be constrained. Conversely, larger k values beyond a certain threshold can potentially distort the original data variations and dilute the influence of genuine collaborators [3].

Therefore, for the rest of our experiments, we use $k = 5$ as it provided the optimal selection of correlated symptoms for imputing a missing symptom for this data.

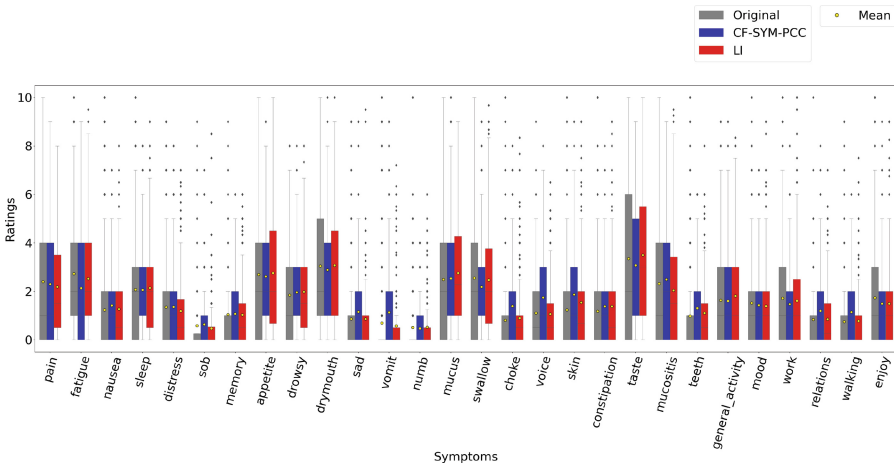


Fig. 4. Box plots of the randomly selected known ratings that were masked for evaluation per symptom. Original represents the pre-substituted ratings while the CF-SYM-PCC and LI represent the predicted or imputed values using the respective techniques.

5.5 Comparing CF-SYM-PCC Against Other Methods

Figure 2 shows the RMSE and MAE comparison between the proposed method (CF-SYM-PCC) and other popular imputation techniques. The results are ordered by descending RMSE and MAE values on the vertical axis, so the best performing method is shown at the bottom.

As can be seen, the CF-SYM-PCC technique was the best performing method with the lowest error rates in both RMSE and MAE metrics (RMSE: 1.447, MAE: 0.898). These results support that leveraging symptom-based collaborative filtering with the Pearson correlation coefficient as the similarity measure is an effective method for PRO data imputation.

The simple Linear Interpolation (LI) method shows the second best comparative performance (RMSE: 1.734, MAE: 1.029) and it is only out-performed by the CF-SYM-PCC method. The good performance of the LI method can be attributed to the fact that symptom ratings are temporally correlated.

Both MICE and KNN imputation had the worst performance with the highest errors (MICE-w-Clinical [RMSE: 2.513, MAE:1.67]), (MICE-Ratings [RMSE: 2.505, MAE: 1.659]), and KNN Imputation (RMSE: 2.123, MAE:1.425). This indicates that clinical features are not very effective in predicting symptom ratings for these patients. There are local correlations between symptoms (e.g. symptom clusters) that using clinical features are not exploited.

The homogeneous nature of the cohort, which received similar treatment regimens for HNC and hence experienced similar symptoms, likely explains the success of the CF method and the relatively limited performance of the clinical data-based methods such as MICE-w-Clinical and KNN. The performance of the techniques may possibly vary under diverse scenarios involving heterogeneous cohorts.

Furthermore, given that symptom ratings often exhibit linearity, it is not surprising that LI and CF-SYM-PCC, which rely heavily on a patient's own ratings, demonstrated relatively superior performance compared to other methodologies.

In the next section, we proceed to compare the performance of the top two methods: LI and CF-SYM-PCC on a per symptom basis.

5.6 Comparing CF-SYM-PCC and LI Techniques Per Symptom

Figure 3 shows the RMSE for the CF-SYM-PCC and LI methods. Each column represents a symptom with two bars. The thicker bar corresponds to the best performing method for that symptom while the thin bar is the RMSE of the other method included for comparison. As can be seen, CF-SYM-PCC had better performance across all the symptoms except for memory, choke and voice. Taste had the highest RMSE (CF-SYM-PCC: 1.909, LI: 2.209) overall, while sadness had the lowest RMSE (CF-SYM-PCC: 0.960, LI: 1.072). Table 4 reports, in addition to the overall RMSE for each symptom and both methods, the per treatment stage RMSE for baseline, during treatment, and after treatment.

Figure 4 demonstrates the spread of both the originally masked and corresponding imputed data, predicted by the CF-SYM-PCC and LI methods. The predictions by both techniques were within the interval of symptom rating values. Also, while both techniques introduced small mean shifts, the distribution of the imputed data is within acceptable ranges from the ground truth, as evidenced by the figure.

Table 4. RMSE per symptom over treatment stages.

| Symptoms | Overall | | Baseline | | During Treatment | | Post Treatment | |
|------------------|------------|-------|------------|-------|------------------|-------|----------------|-------|
| | CF-SYM-PCC | LI | CF-SYM-PCC | LI | CF-SYM-PCC | LI | CF-SYM-PCC | LI |
| Pain | 1.463 | 1.988 | 1.594 | 2.38 | 1.450 | 2.186 | 1.444 | 1.399 |
| Fatigue | 1.542 | 1.861 | 1.966 | 3.235 | 1.446 | 1.691 | 1.604 | 1.767 |
| Nausea | 1.534 | 1.817 | 1.106 | 3.037 | 1.82 | 2.043 | 0.735 | 0.905 |
| Sleep | 1.772 | 2.033 | 1.558 | 2.554 | 1.821 | 1.821 | 1.728 | 2.259 |
| Distress | 1.231 | 1.303 | 1.969 | 1.791 | 1.127 | 1.300 | 1.153 | 1.122 |
| SOB | 1.084 | 1.261 | 1.581 | 2.372 | 0.823 | 0.808 | 1.365 | 1.607 |
| Memory | 1.162 | 1.034 | 1.394 | 0.745 | 1.111 | 0.938 | 1.195 | 1.257 |
| Appetite | 1.681 | 2.154 | 1.398 | 2.576 | 1.705 | 2.116 | 1.706 | 2.100 |
| Drowsy | 1.282 | 1.766 | 1.363 | 2.299 | 1.437 | 1.752 | 0.893 | 1.612 |
| Drymouth | 1.682 | 1.881 | 1.414 | 2.930 | 1.64 | 1.661 | 1.837 | 1.869 |
| Sad | 0.96 | 1.072 | 1.301 | 2.019 | 0.883 | 0.903 | 0.985 | 0.907 |
| Vomit | 1.382 | 1.425 | 0.707 | 1.087 | 1.549 | 1.672 | 1.184 | 0.938 |
| Numb | 1.182 | 1.208 | 0.845 | 0.787 | 0.862 | 0.859 | 1.559 | 1.618 |
| Mucus | 1.56 | 2.214 | 0.816 | 1.751 | 1.430 | 1.887 | 1.818 | 2.681 |
| Swallow | 1.701 | 1.943 | 2.356 | 2.145 | 1.676 | 1.980 | 1.530 | 1.820 |
| Choke | 1.44 | 1.253 | 1.026 | 0.229 | 1.570 | 1.404 | 1.294 | 1.128 |
| Voice | 1.533 | 1.368 | 1.309 | 1.488 | 1.692 | 1.398 | 1.237 | 1.290 |
| Skin | 1.651 | 2.010 | 1.072 | 0.837 | 1.900 | 2.020 | 1.191 | 2.193 |
| Constipation | 1.695 | 1.827 | 1.399 | 2.331 | 1.751 | 1.835 | 1.679 | 1.661 |
| Taste | 1.909 | 2.209 | 1.118 | 3.102 | 1.957 | 2.220 | 1.946 | 1.923 |
| Mucositis | 1.545 | 1.897 | 1.472 | 0.782 | 1.492 | 1.938 | 1.643 | 1.983 |
| Teeth | 1.453 | 1.718 | 1.103 | 1.504 | 1.560 | 1.837 | 1.350 | 1.566 |
| General activity | 1.116 | 1.701 | 1.078 | 1.626 | 1.277 | 1.864 | 0.630 | 1.288 |
| Mood | 1.226 | 1.882 | 1.432 | 2.000 | 1.185 | 1.928 | 1.254 | 1.744 |
| Work | 1.426 | 1.814 | 1.155 | 1.125 | 1.697 | 1.956 | 0.833 | 1.665 |
| Relations | 1.114 | 1.381 | 0.938 | 2.28 | 1.305 | 1.250 | 0.761 | 1.266 |
| Walking | 1.435 | 1.665 | 1.323 | 0.707 | 1.553 | 1.693 | 1.21 | 1.743 |
| Enjoy | 1.4 | 1.957 | 1.704 | 3.338 | 1.433 | 1.885 | 1.244 | 1.606 |

5.7 PCC Correlation Symptom Clusters

Figure 5 is a heat map showing the normalized Pearson correlation coefficients and clustering patterns among symptoms in the CF-PCC-SYM post-imputed dataset. The inter-symptom correlations shown in these figures are computed by averaging the correlations across corresponding time-points for each symptom

pair. These correlation values represent the strength of the relationship between each pair of symptoms. The dendrograms were generated using agglomerative hierarchical clustering and the inter symptom correlation as distance. Initially, each symptom is placed into its own cluster and the highest correlated symptoms are merged first. These clusters are subsequently expanded by averaging the distance between members and other candidate symptoms. The clustering proceeds until all symptoms are in the same cluster.

As can be seen in the figure, there are some strong clusters. These clusters are also evident in the pre-imputed data. These clusters are intuitive and have been identified by prior studies [6, 12, 15, 21, 26]. For example, all interference symptoms {general_activity, walking, work, relations, mood, and enjoy} are clustered together along {distress and sad}. The {appetite, sleep, fatigue, drowsy} is an intuitive cluster, as well as {nausea, vomit} which was strengthened after imputation. The {dry mouth and taste} cluster has also been reported previously together with the {taste, choke, voice, mucus, swallow, pain, and mucositis} cluster [12, 26].

These results show that the CF-SYM-PCC imputed MDASI-HN dataset preserves certain well-established inter-symptom associations or clusters.

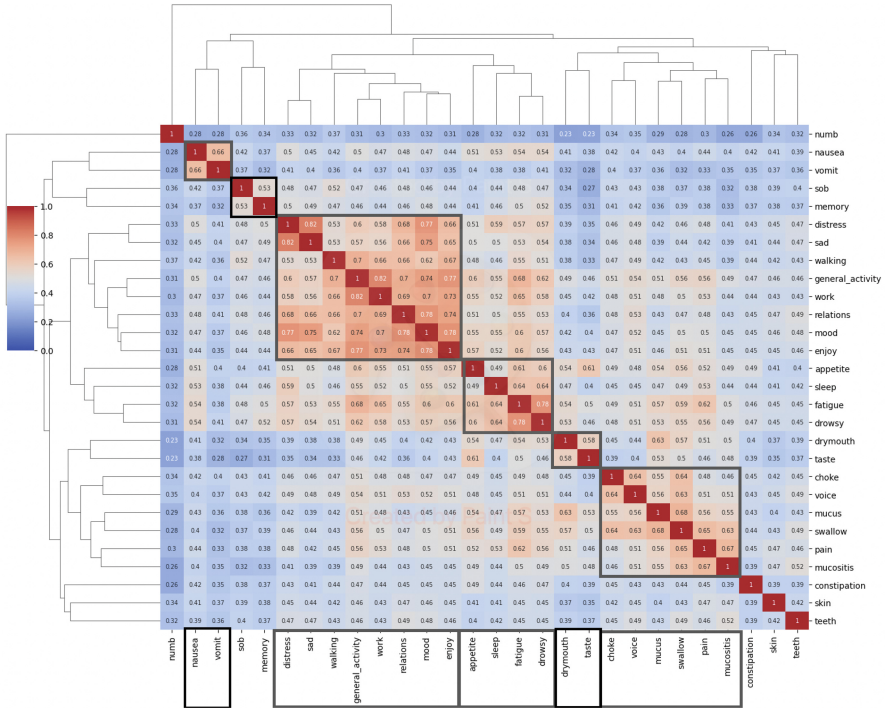


Fig. 5. Average Pearson Correlation Coefficient between every pair of symptoms (after CF imputation). The boxes around the diagonals indicate symptom clusters identified using agglomerative hierarchical clustering. These are consistent with prior literature in symptom cluster analysis and clinically relevant.

6 Conclusion

Collaborative filtering is an effective approach used in recommendation systems to leverage user preferences and recommend items that the user is likely to buy or consume. In this work, we have demonstrated that collaborative filtering can be applied to patient reported outcome data to provide a new and competitive approach for imputing patient data. In our experiments using HNC MDASI-HN data, the best performing configuration of the CF technique was the one denoted as CF-SYM-PCC which use item-based CF and the Pearson Correlation Coefficient for symptom similarity. This CF technique had the best overall (smallest) RMSE and MAE values among all the imputation methods considered, including MICE, KNN imputation, and linear interpolation. Linear interpolation was the second best performing method, and when compared on a per symptom basis, CF-SYM-PCC outperformed Linear Interpolation for 25 out of the 28 symptoms. We partly attribute the excellent performance of the CF method to the homogeneous nature of the cohort, which are all oropharyngeal cancer patients that received similar treatment regimens and hence expected to experience similar symptoms. Evaluating the performance of the CF techniques under diverse scenarios involving heterogeneous cohorts is left as subject for future work.

Acknowledgement. This work was supported directly or in part by funding/resources from the National Institutes of Health (NIH) National Cancer Institute (R01CA258827); received infrastructure support from the MD Anderson Cancer Center Support Grant Head and Neck Cancer Program and Image-Driven Biologically-informed Therapy (IDBT) Program, with programmatic support from the University of Texas MD Anderson Cancer Center Charles and Daneen Stiefel Center for Head and Neck Cancer Oropharyngeal Cancer Research Program; and the MD Anderson Image-guided Cancer Therapy Program.

We further acknowledge Amy C. Moreno, PhD, and Katherine A. Hutcheson, PhD of the STIEFEL program.

References

1. Ayilara, O.F., Zhang, L., Sajobi, T.T., Sawatzky, R., Bohm, E., Lix, L.M.: Impact of missing data on bias and precision when estimating change in patient-reported outcomes from a clinical registry. *Health Qual. Life Outcomes* **17**, 1–9 (2019)
2. Bell, M.L., Fairclough, D.L.: Practical and statistical issues in missing data for longitudinal patient-reported outcomes. *Stat. Methods Med. Res.* **23**(5), 440–459 (2014)
3. Beretta, L., Santaniello, A.: Nearest neighbor imputation algorithms: a critical evaluation. *BMC Med. Inform. Decis. Mak.* **16**, 197–208 (2016)
4. Bhagwat, N., Viviano, J.D., Voineskos, A.N., Chakravarty, M.M., Initiative, A.D.N., et al.: Modeling and prediction of clinical symptom trajectories in Alzheimer’s disease using longitudinal data. *PLoS Comput. Biol.* **14**(9), e1006376 (2018)
5. Caiafa, C.F., Sun, Z., Tanaka, T., Marti-Puig, P., Solé-Casals, J.: Machine learning methods with noisy, incomplete or small datasets (2021)

6. Chiang, S., Ho, K., Wang, S.Y., Lin, C.: Change in symptom clusters in head and neck cancer patients undergoing postoperative radiotherapy: a longitudinal study. *Eur. J. Oncol. Nurs.* **35**, 62–66 (2018)
7. Gangil, T., Shahabuddin, A.B., Dinesh Rao, B., Palanisamy, K., Chakrabarti, B., Sharan, K.: Predicting clinical outcomes of radiotherapy for head and neck squamous cell carcinoma patients using machine learning algorithms. *J. Big Data* **9**(1), 25 (2022)
8. Hodson, T.O.: Root-mean-square error (RMSE) or mean absolute error (MAE): when to use them or not. *Geosci. Model Dev.* **15**(14), 5481–5487 (2022)
9. Jain, G., Mahara, T., Tripathi, K.N.: A survey of similarity measures for collaborative filtering-based recommender system. In: Pant, M., Sharma, T., Verma, O., Singla, R., Sikander, A. (eds.) *Soft Computing: Theories and Applications: Proceedings of SoCTA 2018*, pp. 343–352. Springer, Heidelberg (2020). https://doi.org/10.1007/978-981-15-0751-9_32
10. Koren, Y., Rendle, S., Bell, R.: Advances in collaborative filtering. In: *Recommender Systems Handbook*, pp. 91–142 (2021)
11. van der Laan, H.P., Van den Bosch, L., Schuit, E., Steenbakkers, R.J., van der Schaaf, A., Langendijk, J.A.: Impact of radiation-induced toxicities on quality of life of patients treated for head and neck cancer. *Radiother. Oncol.* **160**, 47–53 (2021)
12. Li, Y., et al.: Symptom clusters in head and neck cancer patients with endotracheal tube: Which symptom clusters are independently associated with health-related quality of life? *Eur. J. Oncol. Nurs.* **48**, 101819 (2020)
13. Luo, Y., Szolovits, P., Dighe, A.S., Baron, J.M.: 3D-mice: integration of cross-sectional and longitudinal imputation for multi-analyte longitudinal clinical data. *J. Am. Med. Inform. Assoc.* **25**(6), 645–653 (2018)
14. , Ma, H., King, I., Lyu, M.R.: Effective missing data prediction for collaborative filtering. In: *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 39–46 (2007)
15. Mathew, A., et al.: Symptom clusters in head and neck cancer: a systematic review and conceptual model. In: *Seminars in Oncology Nursing*, vol. 37, p. 151215. Elsevier (2021)
16. Morton, R.P., Izzard, M.E.: Quality-of-life outcomes in head and neck cancer patients. *World J. Surg.* **27**, 884–889 (2003)
17. Noel, C.W., et al.: Enhancing outpatient symptom management in patients with head and neck cancer: a qualitative analysis. *JAMA Otolaryngol.-Head Neck Surg.* **148**(4), 333–341 (2022)
18. Potdar, K., Pardawala, T.S., Pai, C.D.: A comparative study of categorical variable encoding techniques for neural network classifiers. *Int. J. Comput. Appl.* **175**(4), 7–9 (2017)
19. Raghuwanshi, S.K., Pateriya, R.: Collaborative filtering techniques in recommendation systems. *Data Eng. Appl.* **1**, 11–21 (2019)
20. Rosenthal, D.I., et al.: Measuring head and neck cancer symptom burden: the development and validation of the md Anderson symptom inventory, head and neck module. *Head Neck: J. Sci. Special. Head Neck* **29**(10), 923–931 (2007)
21. Shi, Q., Mendoza, T.R., Gunn, G.B., Wang, X.S., Rosenthal, D.I., Cleland, C.S.: Using group-based trajectory modeling to examine heterogeneity of symptom burden in patients with head and neck cancer undergoing aggressive non-surgical therapy. *Qual. Life Res.* **22**, 2331–2339 (2013)
22. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. *Adv. Artif. Intell.* **2009** (2009)

23. Wang, B.W., Tseng, V.S., et al.: Improving missing-value estimation in microarray data with collaborative filtering based on rough-set theory. *Int. J. Innov. Comput. Inf. Control* **8**(3), 2157–2172 (2012)
24. Wang, Y., et al.: Predicting late symptoms of head and neck cancer treatment using LSTM and patient reported outcomes. In: *Proceedings of the 25th International Database Engineering & Applications Symposium*, pp. 273–279 (2021)
25. Weber, G.M., et al.: Biases introduced by filtering electronic health records for patients with “complete data”. *J. Am. Med. Inform. Assoc.* **24**(6), 1134–1141 (2017)
26. Xiao, C., et al.: Symptom clusters in patients with head and neck cancer receiving concurrent chemoradiotherapy. *Oral Oncol.* **49**(4), 360–366 (2013)



Category-Aware Sequential Recommendation with Time Intervals of Purchases

Jia-Ling Koh^(✉)  and Cheng-Wei Chen

National Taiwan Normal University, Taipei, Taiwan
jlkoh@csie.ntnu.edu.tw

Abstract. The goal of a sequential recommendation system is to predict the next item a user is likely to purchase based on their buying history. Previous research has considered the time intervals between purchases by analyzing patterns in the items, but have neglected the important information at the category level. To overcome this shortcoming, this paper presents two category-aware sequential recommendation models which effectively integrate category information into the user's purchase sequence representation. The first model fuses item embedding with the corresponding category embedding, thus directly infusing category-specific details into the representation of purchasing history, thereby enriching the insight into user behavior. On the other hand, the dual model employs a specialized sub-network to identify patterns within item categories, and this category-level representation indirectly influences the item-level representation of user behavior through an attention mechanism. The results of experiments on Amazon datasets reveal that the inclusion of category data notably improves the hit ratio in sequential recommendation. The proposed models outperform the baseline model particularly in situations involving shorter user sequences. Further, merging purchase records from multiple product datasets across different categories during the training phases leads to even more substantial improvements in the hit ratios.

Keywords: sequence recommendation · category-aware dual model

1 Introduction

Online shopping's ubiquity has skyrocketed in recent years due to technological advances, making product discovery from growing inventories a challenge. Recommendation systems on e-commerce platforms have become pivotal, aiming to offer users tailored product suggestions by analyzing past purchases and user preferences.

Sequential recommendation systems, especially those powered by deep learning methods like CNNs [6, 9] and RNNs [1, 8], are at the forefront of this evolution. While CNNs are good at identifying patterns within continuous purchase sequences, RNNs stand out in handling sequential data across longer durations to provide behavior representation of an entire sequence. Attention mechanisms in neural models can selectively emphasize key items in purchase histories, boosting the accuracy of sequential recommendations [2, 4, 5, 7]. However, a significant oversight in many of these mechanisms is neglecting the time intervals between sequential purchases. Recognizing this

gap, the *TiSASRec* model [3] was developed, incorporating time interval data into a transformer-based recommendation system, resulting in more nuanced predictions.

We propose that integrating item category information into the model can enhance its understanding of purchase interval patterns between different types of products in user behavior. For instance, products with high value or those meant for long-term use often do not require immediate repurchasing. In such cases, the recommendation system should suggest products from a different category that complement the user’s recent purchase; for example, it could recommend peripherals after the user buys a laptop. On the other hand, preference-oriented products may be repurchased in the short term based on user preferences. Hence, the recommendation system should consider both category information and purchase intervals between products in the user’s behavior to provide relevant and suitable recommendations.

To address this issue, this study incorporates both item category information and time intervals between records into a sequential recommendation system. Building upon the *TiSASRec* model [3], two category-aware sequential recommendation models are proposed to effectively integrate category and purchase interval information into the user’s purchase sequence representation. The first model combines the embedding vectors of item IDs and item categories, explicitly integrating category information into the representation of items, thus enhancing the generation of user behavior representation. In contrast, the dual model takes a different approach by employing two separate networks to learn the embedding vectors of time intervals between purchases. It considers the category and item levels in the purchase sequence individually. Subsequently, an attention mechanism is used to combine and leverage information from both levels in the representation of user behavior.

A comparison of our models with *TiSASRec* using the Amazon product datasets shows that incorporating category information effectively enhances the performance of Top-K recommendation. Additionally, both of our proposed models demonstrate significant improvement by merging user purchase records across various categories.

2 Method

2.1 Problem Setup

Each user’s purchase sequence is composed of a series of purchase records. Each record details the item ID, category ID, and purchase timestamp. For ease of reference, let’s define the sets as follows: I represents the set of item IDs, C is the set of item category IDs, and T denotes the set of timestamps corresponding to user purchases.

A purchase record is represented by a triplet (i_k, c_k, t_k) , where $i_k \in I$ is the item ID, $c_k \in C$ is the category ID, and $t_k \in T$ is the purchase timestamp. This record signifies that the user bought item i_k from category c_k at the time t_k . The recent purchase sequence consists of the user’s latest purchase records, sorting chronologically by their purchase timestamps. It takes the form $S_u = \langle (i_1, c_1, t_1), (i_2, c_2, t_2), \dots, (i_L, c_L, t_L) \rangle$. In this sequence, every k^{th} purchase record (where $k = 1, 2, \dots, L$) is represented as $Su_k = (i_k, c_k, t_k)$. Here, L denotes the total number of records in the sequence. The goal is to predict the next potential item that user u is likely to buy for offering a recommendation.

2.2 Category and Time Interval Aware Sequence Recommendation

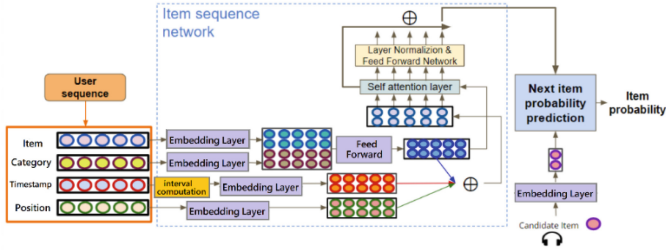


Fig. 1. The model architecture of $TiSASRec^C$.

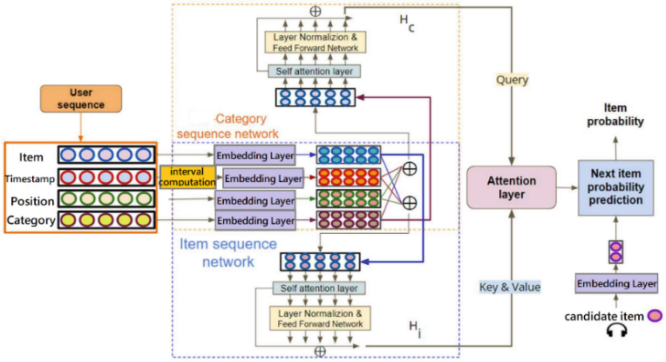


Fig. 2. The model architecture of $TiSASRec^{Dual}$.

We propose two distinct model architectures: $TiSASRec^C$ and $TiSASRec^{Dual}$. Both models are grounded in the foundational $TiSASRec$ model. A self-attention mechanism is deployed to derive the behavioral representation of the purchase sequence for each record. The principal distinction between the models resides in their methodology for fusing item embedding and category embedding.

For the $TiSASRec^C$ model, an immediate fusion of the embedding of item ID and category is undertaken. Conversely, the $TiSASRec^{Dual}$ model first establishes the behavioral representation at the category level from the item categories' purchase sequence. After that, it employs an attention mechanism to estimate the relative importance of each item within the purchase sequence.

Figure 1 illustrates the model architecture of $TiSASRec^C$. This model seamlessly integrates category information, using it as an auxiliary data source to enrich the encoding of each item within the purchase sequence. By combining the embedding of purchase intervals and positions within the sequence with the enhanced item embedding infused with category embedding, the $TiSASRec^C$ model learns item-level purchase features via its Item sequence network. A deeper dive into the details of embedding layers and the self-attention layer in the Item Sequence Network is similar to $TiSASRec$ model [3].

2.3 Framework of the Dual Model

The $TiSASRec^{Dual}$ model adopts a Dual Model framework, where the user’s purchase sequence is learned separately for (1) the category level module and (2) the item level module to capture user behavior features. These modules are represented as the Category sequence network and the Item sequence network, respectively. The model’s architecture is illustrated in Fig. 2.

The design concept of this model is to harness user behavior patterns from both the time intervals between categories in their purchase sequence, leading to more generalized representations, and the time intervals between items, resulting in more specific perspectives. To achieve this, the $TiSASRec^{Dual}$ model employs two distinct sub-networks: Category sequence network and Item sequence network. These sub-networks work independently to learn user behavior features at both the category and item levels. To further consolidate user behavior representations, the model incorporates an attention mechanism. The last user behavior representation $Hc[L]$ from the category level is utilized as the query, while the user behavior representations Hi , obtained after each item purchase, are used as the keys and values. This process effectively calculates the fused user behavior representation, creating a comprehensive and informative final representation for the user.

3 Performance Evaluation

3.1 Dataset Preprocessing and Evaluation Metric

In our experiments, the datasets sourced from Amazon, including a rating-only product review dataset and a product information dataset. The rating-only product review dataset comprises users’ purchase and rating records across various product categories on Amazon from 1998 to 2014. Each record within the dataset includes the User ID, Item ID, Rating, and the Unix-time timestamp of the transaction. On the other hand, the product information dataset details specifics about each product, including the Item ID, Title, Price, Brand, and Categories.

For every user, we chronologically arrange the purchase item based on the purchase timestamp. To determine the category information of each acquired item, we consult the product information dataset, utilizing the Item ID to retrieve the corresponding class hierarchy. The third-tier taxonomy of each item as its category. For example, given an item categorized under the third-tier taxonomy “Sports & Outdoors” -> “Other Sports” -> “Dance”, “Dance” would be identified as that item’s category.

We used five Amazon product datasets, separated into three groups based on various types of purchased items. The first group, comprising higher-priced or longer-lifespan items, includes cellphones and accessories (DB_{Phone}), electronics (DB_{Elect}), and musical instruments (DB_{Music}). The second group, daily necessities, includes home and kitchen products (DB_{Home}). The final group, preference-oriented items possible with irregular purchasing intervals, includes clothing, shoes, and jewelry (DB_{Cloth}). We set the maximum length of user sequences, denoted as L , as 50. If a user’s history had fewer than L records, we padded the sequence with empty records at the start. Table 1 displays the statistics for each dataset, including the number of users, the number of items, the

number of categories, the average number of purchase records per user, *Sparsity*, and the average sequence length per user. During the training phase, we assigned the last item each user purchased as the prediction target for the test data. Besides, the second-to-last item served as the prediction target for the validation data. We constructed the training dataset using all the other previous purchase records. Table 2 enumerates the total number of user purchase records within the training, validation, and test datasets.

Table 1. Statistics of the product datasets

| Dataset | #User | #Item | #Category | #Purchase | Sparsity | Avg (seg_len) |
|---------------------------|---------|---------|-----------|-----------|-----------------------|---------------|
| <i>DB_{Phone}</i> | 65,876 | 59,759 | 133 | 285,239 | 7.25×10^{-5} | 4.33 |
| <i>DB_{Elect}</i> | 247,287 | 143,040 | 164 | 1,557,908 | 4.4×10^{-5} | 6.3 |
| <i>DB_{Music}</i> | 8,799 | 13,247 | 72 | 37,924 | 3.25×10^{-4} | 4.31 |
| <i>DB_{Home}</i> | 111,373 | 92,626 | 123 | 614,779 | 5.6×10^{-5} | 5.52 |
| <i>DB_{Cloth}</i> | 166,047 | 174,278 | 1,048 | 665,848 | 2.3×10^{-5} | 4.01 |

Table 2. The number of purchase records in the training, validation, and test sets.

| Dataset | #purchase (Train) | #purchase (Validation) | #purchase (Test) |
|---------------------------|-------------------|------------------------|------------------|
| <i>DB_{Phone}</i> | 153,489 | 65,875 | 65,875 |
| <i>DB_{Elect}</i> | 1,063,334 | 247,287 | 247,287 |
| <i>DB_{Music}</i> | 20,326 | 8,799 | 8,799 |
| <i>DB_{Home}</i> | 392,033 | 111,373 | 111,373 |
| <i>DB_{Cloth}</i> | 333,754 | 166,047 | 166,047 |

In the following experiments, we evaluate recommendation performance using two metrics: Hit Ratio at k ($HR@k$) and Normalized Discounted Cumulative Gain ($NDCG$). The evaluation process involves test data that includes the correct target item and 100 randomly sampled negative samples. These items are then ranked based on their predicted probabilities. In the experiments, the values of k are set as 5 and 10.

3.2 Results of Performance Evaluation

Experiment 1. In this experiment, we compare the performance of the $TiSASRec^C$ and $TiSASRec^{Dual}$ models against the baseline model, $TiSASRec$ [3]. Table 3 illustrates the average hit rates and $NDCG$ values of the three models. The best results of the evaluation metrics on each dataset are bold underlined. Except for the *DB_{Phone}* dataset, the $TiSASRec^C$ model consistently surpasses the $TiSASRec$ model in $HR@5$, $HR@10$, and various $NDCG$ values across the other four product datasets. This indicates that incorporating product category information from purchase records can enhance the effectiveness of the next item recommendation. Notably, while the $TiSASRec^{Dual}$ model doesn't

always outperform the others, it exhibits the best performance in terms of $HR@10$ and $NDCG@5$ on the DB_{Music} dataset.

Table 3. The HR and $NDCG$ evaluation results for the three models.

| Dataset | $TiSASRec$ | $TiSASRec^C$ | $TiSASRec^{Dual}$ | $TiSASRec$ | $TiSASRec^C$ | $TiSASRec^{Dual}$ |
|--------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|
| | HR@5 | | | HR@10 | | |
| DB_{Phone} | <u>0.4761</u> | 0.4636 | 0.3640 | 0.5635 | <u>0.5534</u> | 0.4759 |
| DB_{Elect} | 0.5214 | <u>0.5300</u> | 0.5105 | 0.6295 | <u>0.6350</u> | 0.6220 |
| DB_{Music} | 0.2656 | <u>0.3180</u> | 0.3166 | 0.3619 | 0.4330 | <u>0.4373</u> |
| DB_{Home} | 0.3663 | <u>0.3697</u> | 0.3458 | 0.4814 | <u>0.4853</u> | 0.4696 |
| DB_{Cloth} | 0.2968 | <u>0.3107</u> | 0.2759 | 0.3868 | <u>0.4033</u> | 0.3739 |
| | NDCG@5 | | | NDCG@10 | | |
| DB_{Phone} | <u>0.3654</u> | 0.3573 | 0.2661 | <u>0.2434</u> | 0.2337 | 0.1552 |
| DB_{Elect} | 0.4039 | <u>0.4051</u> | 0.3883 | 0.2671 | <u>0.2675</u> | 0.2531 |
| DB_{Music} | 0.1837 | <u>0.2247</u> | <u>0.2247</u> | 0.1035 | <u>0.1301</u> | 0.1272 |
| DB_{Home} | 0.2573 | <u>0.2615</u> | 0.2444 | 0.1422 | <u>0.1449</u> | 0.1356 |
| DB_{Cloth} | 0.2147 | <u>0.2257</u> | 0.1961 | 0.1277 | <u>0.1352</u> | 0.1126 |

Table 4. The $HR@10$ evaluation results on the popular/unpopular category items.

| Dataset | Popular% | Popular category items | | | Unpopular category items | | |
|--------------|----------|------------------------|----------------------|-------------------|--------------------------|----------------------|----------------------|
| | | $TiSASRec$ | $TiSASRec^C$ | $TiSASRec^{Dual}$ | $TiSASRec$ | $TiSASRec^C$ | $TiSASRec^{Dual}$ |
| DB_{Phone} | 46.93% | <u>0.5215</u> | 0.5115 | 0.4128 | 0.5926 | <u>0.5952</u> | 0.5411 |
| DB_{Elect} | 38.19% | <u>0.6797</u> | 0.6791 | 0.6695 | 0.6091 | <u>0.6091</u> | 0.5940 |
| DB_{Music} | 38.29% | 0.4357 | <u>0.4960</u> | 0.4924 | 0.3862 | 0.3862 | <u>0.4011</u> |
| DB_{Home} | 25.39% | <u>0.5625</u> | 0.5593 | 0.5490 | 0.4588 | <u>0.4588</u> | 0.4426 |
| DB_{Cloth} | 45.25% | 0.4148 | <u>0.4209</u> | 0.3903 | 0.3837 | <u>0.3837</u> | 0.3607 |

Experiment 2. To assess the influence of category occurrence frequency on the prediction performance of the models, we label categories accounting for 10% or more of the purchase records in a dataset as “popular categories”. We partition the test data into two distinct groups: the first group includes items from popular categories (termed “Popular category items”), while the second group is made up of items from non-popular categories (denoted as “Unpopular category items”). It’s important to highlight that, even when predicting for popular categories, the models do not eliminate the chance of predicting items from non-popular categories.

In Table 4, the proportion of popular items within the test data is displayed, accompanied by the $HR@10$ outcomes for each model across both test data groups. The findings reveal that both the $TiSASRec^C$ and $TiSASRec^{Dual}$ models outperform the $TiSASRec$

model when predicting items from non-popular categories. This improvement is particularly prominent in the DB_{Music} and DB_{Cloth} datasets. Such results demonstrate that the two proposed models don't merely focus on predicting items from popular categories. Instead, they effectively leverage the patterns discerned at the category level to enhance predictions.

Experiment 3. This experiment evaluates the potential benefits of integrating purchase data from a secondary dataset to enhance a model's recommendation capabilities for a primary dataset.

Three distinct datasets were chosen for this analysis: DB_{Elect} , DB_{Home} , and DB_{Cloth} , each demonstrating unique characteristics of purchase interval. To distinguish the datasets by integrating two datasets, the single datasets are labeled as sDB_{Elect}^{Single} , sDB_{Home}^{Single} , and sDB_{Cloth}^{Single} , respectively. Furthermore, hybrid datasets combining records from one dataset (C') to predict items in another dataset (C) were generated, termed $DB_{C\&C'}^{Combine}$, with C and C' being any two distinct datasets from the initial three.

Table 5 details the $HR@10$ outcomes for the three models across the nine datasets, emphasizing the best results in bold underline. The column "mixing %" showcases the proportion of users present in both the merged datasets in relation to the primary dataset. By merging dataset of various categories, Table 5 shows a consistent prediction performance enhancement across all the three models. Notably, both $TiSASRec^C$ and $TiSASRec^{Dual}$ exhibit more substantial improvements compared to $TiSASRec$. The table also highlights that, for sDB_{Elect}^{Single} and sDB_{Home}^{Single} datasets, $TiSASRec^{Dual}$ realizes the most enhancements when merging datasets. In contrast, for sDB_{Cloth}^{Single} , $TiSASRec^C$ shows the most marked improvement drawing from results of $DB_{Cloth\&Elect}^{Combine}$. This indicates that by merging different datasets, both $TiSASRec^{Dual}$ and $TiSASRec^C$ can discover patterns across a sequence of more diverse categories and temporal intervals. Moreover, when

Table 5. The $HR@10$ and improving rate gained from the hybrid datasets.

| Dataset | mixing% | $TiSASRec$ | | $TiSASRec^C$ | | $TiSASRec^{Dual}$ | |
|-------------------------------|---------|------------|---------|----------------------|----------------------|----------------------|-----------------------|
| sDB_{Elect}^{Single} | | 0.3787 | | <u>0.4060</u> | | 0.3894 | |
| $DB_{Elect\&Cloth}^{Combine}$ | 25.81% | 0.3992 | +5.42% | 0.4455 | +9.73% | <u>0.4457</u> | <u>+14.45%</u> |
| $DB_{Elect\&Home}^{Combine}$ | 39.17% | 0.4010 | +5.89% | 0.4260 | +4.92% | <u>0.4340</u> | <u>+11.44%</u> |
| sDB_{Home}^{Single} | | 0.2611 | | <u>0.2616</u> | | 0.2423 | |
| $DB_{Home\&Cloth}^{Combine}$ | 47.76% | 0.2890 | +10.67% | 0.2979 | +13.87% | <u>0.3111</u> | <u>+28.40%</u> |
| $DB_{Home\&Elect}^{Combine}$ | 60.96% | 0.2783 | +6.57% | 0.3074 | +17.50% | <u>0.3154</u> | <u>+30.17%</u> |
| sDB_{Cloth}^{Single} | | 0.1110 | | <u>0.1395</u> | | 0.1309 | |
| $DB_{Cloth\&Elect}^{Combine}$ | 47.69% | 0.1318 | +18.73% | <u>0.1680</u> | <u>20.41%</u> | 0.1512 | +15.47% |
| $DB_{Cloth\&Home}^{Combine}$ | 48.13% | 0.1114 | +0.35% | 0.1481 | +6.14% | <u>0.1518</u> | <u>+15.93%</u> |

fusing datasets that capture long-term and short-term purchase tendencies between categories, this positions the $TiSASRec^{Dual}$ model in a stronger vantage point to predict a user's next purchase item.

4 Conclusion

In this paper, we dive deep into the domain of sequential recommendation systems, introducing two models: $TiSASRec^C$ and $TiSASRec^{Dual}$. These models leverage categorical embeddings to enhance the representation learning of user behaviors and temporal purchase patterns. The evaluations highlight that the $TiSASRec^C$ model, enriched with direct categorical embeddings, is particularly adept at handling dense, long-term purchase histories. On the other hand, the $TiSASRec^{Dual}$ model, which employs a more implicit category-driven feature extraction, stands out when dealing with sparse data or when category purchase frequencies are relatively balanced in the dataset. Moreover, an innovative aspect of $TiSASRec^{Dual}$ is its capability to fuse purchase data from dual categories, enabling a more granular extraction of inter-category purchase intervals. When benchmarked against the result that processes purchase sequences from a single-category dataset, both of our proposed architectures exhibit significant performance gains when applied to hybrid datasets. For future work, we suggest integrating the recommendation time as an added input. This methodology would empower the model to deliver context-aware recommendations aligned with varying purchase times, thus addressing the diverse preferences and needs of users.

References

1. Bai, T., et al.: CTRec: a long-short demands evolution model for continuous-time recommendation. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2019), pp. 675–684 (2019)
2. Kang, W.C., McAuley, J.: Self-attentive sequential recommendation. In: Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM), pp. 197–206 (2018)
3. Li, J., Wang, Y., McAuley, J.: Time interval aware self-attention for sequential recommendation. In: Proceedings of the 13th ACM International Conference on Web Search and Data Mining (WSDM 2020), pp. 322–330 (2020)
4. Li, Y., Chen, T., Zhang, P.F., Yin, H.: Lightweight self-attentive sequential recommendation. In: Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM 2021), pp. 967–977 (2021)
5. Lin, J., Pan, W., Ming, Z.: FISSA: fusing item similarity models with self-attention networks for sequential recommendation. In: Proceedings of the 14th ACM Conference on Recommender Systems, pp. 130–139 (2020)
6. Tang, J., Wang, K.: Personalized top-n sequential recommendation via convolutional sequence embedding. In: Proceedings of the 11th ACM International Conference on Web Search and Data Mining (WSDM 2018), pp. 565–573 (2018)
7. Wang, J., Liu, Q., Liu, Z., Wu, S.: Towards accurate and interpretable sequential prediction: a CNN and attention-based feature extractor. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM 2019), pp. 1703–1712 (2019)

8. Yu, F., Liu, Q., Wu, S., Wang, L., Tan, T.: A dynamic recurrent model for next basket recommendation. In: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR 2016), pp. 729–732 (2016)
9. Yuan, F., Karatzoglou, A., Arapakis, I., Jose, J.M., He, X.: A simple convolutional generative network for next item recommendation. In: Proceedings of the 12th ACM International Conference on Web Search and Data Mining (WSDM 2019), pp. 582–590 (2019)



A Soft Actor-Critic Algorithm for Sequential Recommendation

Hyejin Hong^(✉), Yusuke Kimura^{}, and Kenji Hatano^{}

Doshisha University, Kyoto, Japan
{hong,kimura,hatano}@mil.doshisha.ac.jp

Abstract. Recently, it has become common knowledge that using reinforcement learning for a sequential recommendation, which predicts a user's next action, can improve recommendation performance. This is because reinforcement learning can be used to efficiently learn behavioral changes, which can help you better understand user behavior patterns. Previous research has attempted to incorporate dynamic user characteristics through Actor-Critic algorithms, but these methods are limited in their ability to adequately learn user behavior because they learn without distinguishing between past and present behavior. Therefore, in this study, we propose a framework that incorporates the SAC algorithm, a reinforcement learning technique, to identify correlations between users and items in a dynamic environment where the recommender system continuously receives the next time series of data. Our framework outperformed from the viewpoint of the accuracy in the recommender system compared with the existing methods, and we could confirm that the SAC algorithm has the potential to improve the quality of the sequential recommendations in capturing the temporal dynamics of user interactions.

Keywords: Sequential recommendation · Reinforcement learning · Recommender system · Multi-task learning

1 Introduction

Recommender systems are widely used across various domains, and this research area is evolving rapidly. These systems alleviate the burden of filtering vast amounts of information by recommending relevant content to users, not only on e-commerce (EC) sites but also for wayfinding and tourist attraction recommendations. Understanding user behavior patterns is crucial for enhancing the accuracy of recommender systems from various viewpoints, as analyzing these patterns allows for the extraction of user characteristics, enabling more accurate, personalized recommendations [11].

The widespread collection of sequential log data captures users' behavior sequentially and is invaluable for sequential recommender systems [8]. The characteristic of the trial-and-error learning of various patterns and the rapid adaptation to the new status of time series data can adapt to user preferences, providing a recommendation mechanism based on learned behavior changes.

In recommender systems, deep learning has gained popularity for its ability to identify complex and non-linear relationships between users and items, providing cutting-edge performance. However, it faces challenges such as being data-hungry, and computationally expensive [17]. Reinforcement learning addresses these issues by allowing agents to learn from environmental rewards without extensive training data, making it highly effective for recommender systems [1].

Reinforcement learning, especially the Actor-Critic (AC) algorithms, is gaining prominence in sequential recommendations. Because AC algorithms adapt to dynamic environments, they have been shown to learn well in complex environments such as recommender systems. They are beneficial in capturing prolonged interactions between users and items [9]. However, applying the AC algorithms to recommender systems can be challenging due to common issues like overestimating state values and the inability to update past actions. These issues can prevent the system from adapting to changing user preferences [16]. The problem of overestimating the state values of the AC algorithm can also occur in sequential recommendation, which is a type of recommendation [18].

To cope with these issues, we propose a framework using the Soft Actor-Critic (SAC) algorithm [5], adjusted for the sequential recommendation framework, to update evaluations of past actions and learn various behavioral patterns of users. In addition, our method addresses the problem of overestimating the AC algorithm by passing the output values of supervised learning models trained on pre-collected data to the SAC algorithm to enable stable learning of the reinforcement learning model. In environments like EC sites, where we must consider various factors, we utilize multi-task learning, which enables simultaneous learning of multiple tasks to enhance the performance of recommender systems. Our framework aims to leverage the complexity of such environments to improve the accuracy of recommendations through learning user behavior prediction tasks.

2 Related Work

To optimize the output from recommender systems, we must consider each user's unique behavioral patterns [15]. These systems scrutinize sequential user data to understand individual preferences, but we must take into account many factors to interpret these preferences accurately.

Old-fashioned methods, which categorize users by personal attributes such as gender or age and to recommend items based on social stereotypes constructed from these categories, cannot provide personalized recommendations [13]. Therefore, the prior researches in this field have segmented user categorization into more extensive categories using other factors that can be considered for personalization, such as the user's personality [14].

However, previous research described above has demonstrated that such an approach needs to be revised to personalize recommendations. Consequently, the recommender systems that assess users' nuanced, ongoing behavioral patterns were drawn to attention. Such recommender systems offer a more precise identification of individual preferences by analyzing user interest patterns and

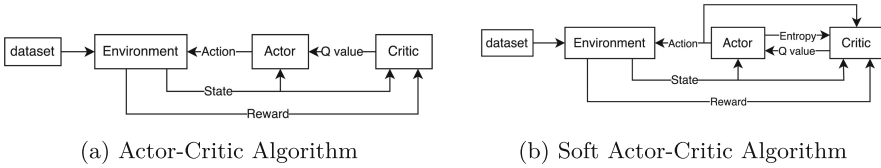


Fig. 1. Structures of the AC and SAC algorithms.

behaviors derived from their continuous behavioral data. That is to say, they need to grasp the sequence of user actions through sequential data is essential to interpret these continuous behavioral traits accurately. Thus, they are called sequential recommender systems and tend to adopt reinforcement learning to achieve this goal.

2.1 Multi-task Learning

Reinforcement learning needs help in applying the synergies of learning multiple tasks together. Therefore, multi-task learning should be adopted to address multiple prediction tasks in a recommender system [8]. Multi-task learning is a methodology that facilitates learning common features and patterns across tasks despite each task having different objectives. The prediction tasks in recommender systems are typically forecasting items that will be clicked (Click-Through Rate; CTR), predicting which items will be purchased (Conversion Rate; CVR), and estimating the conversion rate from clicks to purchases (Click-Through Conversion Rate; CTCVR).

However, when using multi-task learning, increasing the prediction accuracy of one task to the detriment of another leads to task interference, a phenomenon known as task overlap. Recent studies have highlighted the importance of addressing task interference in multi-task learning applications, and advances in gradient surgery techniques have been proposed [6].

2.2 TD3 in RMTL

Reinforcement learning enhanced Multi-Task Learning (RMTL) framework integrates reinforcement learning to adjust weights within the multi-tasking system dynamically, enhancing recommendation accuracy [10]. To solve the problem that existing recommendation models are built based on itemized datasets and ignore the patterns of interaction between users and items, we used the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm [4] to efficiently identify user behavior features of sequential data and optimize the loss function of multi-tasks. RMTL leverages reinforcement learning’s trial-and-error nature and dynamic weights to address multi-tasking challenges, improving both the multi-tasking framework and recommendation performance.

Using the TD3 algorithm, RMTL addresses task interference but struggles with changing user preferences and diverse item attributes [2]. We have to

develop an environment that allows for more unconstrained learning to address the TD3 algorithm’s limitations in learning various actions within a recommendation environment.

2.3 Soft Actor-Critic Algorithm

The SAC algorithm offers superior sampling efficiency and stability over other AC algorithms by incorporating an entropy-maximizing term [5]. In an environment where a probability distribution represents the following action for a state, the SAC algorithm selects the optimal action regardless of the current strategy. It adopts on-policy and off-policy learning to realize high-efficiency sampling. It also stabilizes the learning of reinforcement learning models by considering an entropy maximization term in the objective function of the AC algorithm. By comparing Figs. 1a and 1b, it can be observed that SAC transfers entropy from the actor to the critic. As a result, SAC is less sensitive to hyperparameters, ensuring stability in environments that require training with different random seeds and less data. SAC’s learning objective is to explore new states by making diverse choices in a broader range of environments with higher rewards. This approach would not only mitigate the identified shortcomings of the TD3 algorithm but also be supposed to enhance the overall performance and reliability of multi-task learning frameworks in complex recommendation scenarios.

However, the AC algorithms, including SAC and TD3, tend to overestimate values during learning, a challenge that persists despite using target networks to stabilize predictions [3].

3 Methodology

Our methodology leverages the SAC algorithm to address the overestimation of value predictions commonly observed in reinforcement learning-based recommender systems. By integrating supervised predictions, our framework improves the stability and accuracy of recommendations.

The key components that underpin our methodology in this study include:

- The SAC algorithm’s adaptability allows our system to respond to diverse user needs and environmental changes effectively.
- It handles hyperparameter variations crucial for optimal performance across different recommendation tasks and datasets.
- Our approach mitigates common overestimation issues in reinforcement learning, enhancing the learning process’s stability.

3.1 Our Framework

The integration of reinforcement learning with the recommender system begins with the actor predicting user actions based on the current state, as shown in Fig. 2. Our multi-task learning model, pre-trained on metrics like CTR and CTCVR, assists in making accurate predictions.

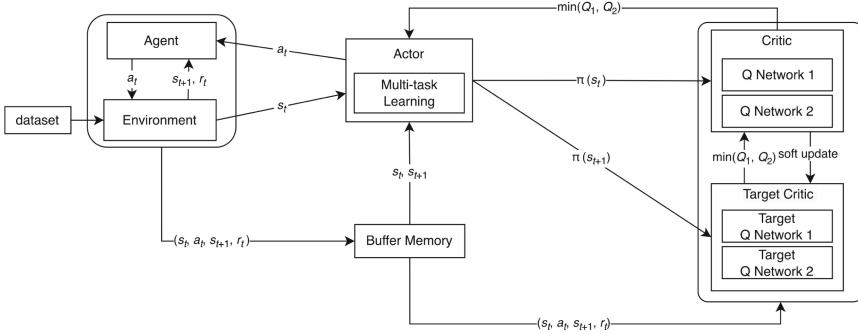


Fig. 2. Our framework for a multi-task recommender system using the SAC reinforcement learning algorithm.

The agent chooses an action a_t and sends it to the environment, which sends the values of the next state and reward to the agent. At the same time, it sends information about the current state s_t , action policy a_t , and the next state s_{t+1} and reward r_t to the buffer memory. The actor receives the information about the current state s_t and the next state s_{t+1} from the buffer memory and sends the respective values of the current state $\pi(s_t)$ and the next state $\pi(s_{t+1})$ to the critic and the target critic, respectively, based on the multi-task learning results. The next state $\pi(s_{t+1})$ is chosen based on the predictions inferred by the multi-task learning model, which is expected to reduce the uncertainty in the predictions of reinforcement learning.

The critic then calculates the value based on the current state and sends a soft update to the target critic, who calculates a value based on information about the $\pi(s_{t+1})$ and sends the smaller value of the two target critics $\min(Q_1, Q_2)$ to the critic. The critic sends this value to the actor for multi-task learning. Repeating the above process builds learning results for various patterns in steps.

3.2 Actor

In our methodology, the actor network uses multi-task learning to generate predictions of user behavior that align with the system’s multiple goals. The actor learns to model complex user interactions and preferences to predict actions that maximize the expected reward over user sessions. Actors also send the results from multi-task learning to the Critic Network and Target Critic Network, which use them to calculate value. By leveraging the features of the multi-task that predict the user’s next behavior, the transferred values are used to adjust the values of the target critic to reduce overestimation.

3.3 Critic

The critic networks in our multi-task recommender system serve to evaluate the quality of Q_1 and Q_2 , each responsible for one aspect of the multi-task objective:

evaluating actions concerning their CTR and CTCVR prediction performance. These networks provide a mechanism for judging the potential long-term rewards of actions across different user sessions and adjusting the actor’s policy toward more profitable recommendations.

$$y_i = r_i + \gamma \min (Q_{\theta'_1}(s_{t+1}, \pi_{\phi'}(s_{t+1})), Q_{\theta'_2}(s_{t+1}, \pi_{\phi'}(s_{t+1}))) - \alpha \log \pi_{\phi'}(a_i | s_{t+1}) \quad (1)$$

In Eq. 1, y_i represents the target value for each action, incorporating the r_i , the discounted future reward from the s_{t+1} , and a penalty term according to the policy’s entropy for action i , where $i = 1, 2$. The policy function $\pi_{\phi}(s)$, parameterized by the actor random parameters ϕ , maps s to a distribution over possible actions. γ is the discount factor that weights the importance of future rewards in the calculation of the target value, and $\alpha \log \pi_{\phi'}(a_i | s_{t+1})$ is the entropy adjustment term for action i , penalizing low entropy (high certainty) in action selection to encourage exploration. The entropy coefficient α is dynamically adjusted to maintain a balance between exploration and exploitation, which is particularly crucial in a recommender system where user preferences and item relevance may shift over time.

$$L(\theta_i) = \frac{1}{N} \sum (y_i - Q_{\theta_i}(s_t, a_i))^2 \quad (2)$$

In Eq. 2, $L(\theta_i)$ is the loss function of critic network i , measuring the discrepancy between the target value y_i and the predicted Q-value $Q_{\theta_i}(s_t, a_i)$.

Through this mechanism, the critic networks guide the actor toward a policy attuned to the dynamics of user behavior as reflected in the session data.

4 Experiments and Discussions

Our method’s objective is to leverage the features of the SAC algorithm to propose a framework for a sequential recommender system that positively influences recommendation performance. We conducted evaluation experiments to verify the effectiveness of this framework, focusing on the evolution of prediction performance in multi-task learning of reinforcement learning and the impact of the variation of the loss (a-loss) incurred by the actor-network undergoing multi-task learning on the quality of recommendations. In this study, as evaluation indices, we use Area Under the ROC Curve (AUC), which confirms the performance of the ranking task of the recommender system, and Logloss, which is the certainty of the prediction, to confirm the performance of the prediction task.

Incidentally, our method employs linear scalarization (LS) [7], which performs best on validation data, to adjust for task interference in multi-task learning, which differs from the RMTL approach. Therefore, we investigate the impact of our method by comparing the evaluation metrics of RMTL-LS, which is a reformulation of the existing method, RMTL, with the proposed method.

Table 1. Results of an evaluation experiment: Bold type denotes the results of the method with the best performance in each evaluation index.

| | RMTL | RMTL-LS | Our Method |
|----------|-------|---------|--------------|
| a-loss ↓ | 2.753 | 2.740 | 1.577 |

Table 2. As a result of the performance of the tasks used in multi-task training, which is used as an evaluation metric for recommender systems.

| | RMTL | | RMTL-LS | | Our Method | |
|-------------|--------|--------|---------|--------|---------------|---------------|
| | CTR | CVCTR | CTR | CVCTR | CTR | CVCTR |
| AUC ↑ | 0.7263 | 0.7300 | 0.7275 | 0.7306 | 0.7318 | 0.7379 |
| Logloss ↓ | 0.2060 | 0.0486 | 0.2060 | 0.0490 | 0.2018 | 0.0481 |
| s-Logloss ↓ | 0.0842 | 0.0151 | 0.0839 | 0.0150 | 0.0840 | 0.0149 |

Using the Retailrocket recommender system dataset¹ on Kaggle Web site, which includes user behavior data from e-commerce sites, we employed an existing multi-task learning framework for recommendation systems, the Entire Space Multi-task Model (ESMM) [12], which is specialized for CTR.

The results showed that our method not only achieved the lowest a-loss but also enhanced prediction accuracy for individual tasks, as indicated in our results tables (see Tables 1 and 2).

The enhancement in recommendation performance is attributed to two factors. First, the SAC algorithm’s ability to learn from a broader range of user situations, represented by a probability distribution of next actions, allows for a wider variety of recommendation scenarios by actively selecting diverse actions. Second, our method of sharing the next state obtained as a result of the actor with the target critic can mitigate overestimation, potentially reducing q-loss.

5 Conclusion

This paper proposes a framework for utilizing the SAC algorithm to enhance recommendation accuracy in a sequential recommender system. In particular, we tailored the SAC algorithm to develop the sequential recommender system. We constructed a framework that shares the next state of time series data to address the overestimation issue inherent in the AC algorithm. The primary challenge of this research is to construct a model capable of offering novel recommendations to users while ensuring stable learning. Our experimental evaluation indicated that multi-task learning improved recommendation accuracy and could effectively predict user behavior compared to the existing methods. Therefore, we confirmed

¹ Retailrocket recommender system dataset, <https://www.kaggle.com/datasets/retailrocket/ecommerce-dataset>, last accessed: August 5, 2024.

the effectiveness of our framework with the SAC algorithm and its function of reducing task interference.

Shortly, we would like to treat the following considerable points:

- We want to adopt our method into the broader field of recommender systems, such as point-of-interest (POI) recommender systems that suggest the next place for a user to visit.
- Applying the SAC algorithm tailored to recommender systems increased the computational cost of our method. Therefore, it is necessary to develop an efficient update method by analyzing the parts that have a high computational cost, such as updating target critics.

Acknowledgements. This work was partly supported by the Grants-in-Aid for Academic Promotion, Graduate School of Culture and Information Science, Doshisha University, JSPS KAKENHI Grant Number JP23K28383 and JP24K03044.











References

1. Afsar, M.M., Crump, T., Far, B.: Reinforcement learning based recommender systems: a survey. *ACM Comput. Surv.* **55**(7) (2022)
2. Bangari, S., et al.: A review on reinforcement learning based news recommendation systems and its challenges. In: *Proceedings of 2021 International Conference on Artificial Intelligence and Smart Systems*, pp. 260–265 (2021)
3. Barakat, A., Bianchi, P., Lehmann, J.: Analysis of a target-based actor-critic algorithm with linear function approximation. In: *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, pp. 991–1040. PMLR (2022)
4. Fujimoto, S., et al.: Addressing function approximation error in actor-critic methods. In: *Proceedings of the 35th International Conference on Machine Learning*, vol. 80 (2018)
5. Haarnoja, T., et al.: soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: *Proceedings of the 35th International Conference on Machine Learning*, vol. 80 (2018)
6. Kumar, A., Daumé, H.: Learning task grouping and overlap in multi-task learning. In: *Proceedings of the 29th International Conference on International Conference on Machine Learning*, pp. 1723–1730. Omnipress (2012)
7. Lin, X., et al.: Pareto multi-task learning. In: *Thirty-third Conference on Neural Information Processing Systems (NeurIPS)*, pp. 12037–12047 (2019)
8. Lin, Y., et al.: A survey on reinforcement learning for recommender systems. *IEEE Trans. Neural Netw. Learn. Syst.* 1–21 (2023)
9. Liu, Z., Wen, S., Quan, Y.: Deep reinforcement learning based group recommender system. *arXiv preprint [arXiv:2106.06900](https://arxiv.org/abs/2106.06900)* (2021)
10. Liu, Z., et al.: Multi-task recommendations with reinforcement learning. In: *Proceedings of the ACM Web Conference 2023*, pp. 1273–1282 (2023)
11. Ma, J., et al.: Off-policy learning in two-stage recommender systems. In: *Proceedings of The Web Conference 2020*, pp. 463–473. ACM (2020)
12. Ma, X., et al.: Entire space multi-task model: an effective approach for estimating post-click conversion rate. In: *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 1137–1140 (2018)

13. Shani, G., Meisles, A., Gleyzer, Y., Rokach, L., Ben-Shimon, D.: a stereotypes-based hybrid recommender system for media items. In: Proceedings of AAAI Workshop on Intelligent Techniques for Web Personalization, pp. 76–83 (2007)
14. Sohail, S.S., Siddiqui, J., Ali, R.: Classifications of recommender systems: a review. *J. Eng. Sci. Technol. Rev.* **10**(4) (2017)
15. Wu, W., Chen, L., Zhao, Y.: Personalizing recommendation diversity based on user personality. *User Model. User-Adap. Inter.* **28**(3), 237–276 (2018)
16. Xin, X., et al.: Self-supervised reinforcement learning for recommender systems. In: Proceedings of the 43rd International ACM SIGIR Conference on Research & Development in Information Retrieval, pp. 931–940 (2020)
17. Zhang, S., Yao, L., Sun, A., Tay, Y.: Deep learning based recommender system: a survey and new perspectives. *ACM Comput. Surv.* **52**(1) (2019)
18. Zhao, L.Y., et al.: An overestimation reduction method based on the multi-step weighted double estimation using value-decomposition multi-agent reinforcement learning. *Neural Process. Lett.* **56**(3), 1–21 (2024)



An Approach for Social-Distance Preserving Location-Aware Recommender Systems: A Use Case in a Hospital Environment

Marcos Caballero¹, María del Carmen Rodríguez-Hernández¹,
Raúl Parada², Sergio Ilarri³, Raquel Trillo-Lado³,
Ramón Hermoso⁴, and Óscar J. Rubio⁵

¹ Technological Institute of Aragón (ITA), María de Luna 7-8, Zaragoza, Spain
{[mcaballero](mailto:mcaballero@itainnova.es),[mcrodriguez](mailto:mcrodriguez@itainnova.es)}@itainnova.es

² Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA),
Av. Carl Friedrich Gauss 7, Castelldefels, Barcelona, Spain
rparada@cttc.cat

³ I3A, Universidad de Zaragoza, María de Luna 1, Zaragoza, Spain
{[silarri](mailto:silarri@unizar.es),[raqueltl](mailto:raqueltl@unizar.es)}@unizar.es

⁴ Universidad de Zaragoza, Violante de Hungría 23, Zaragoza, Spain
rhermoso@unizar.es

⁵ Imascono Company, Josefa Amar y Borbón 10, Zaragoza, Spain
oscar@imascono.com

Abstract. Currently, the volume of geo-referenced data is rapidly expanding, and users frequently show interest in nearby items. Consequently, Location-Aware Recommender Systems (LARS) have garnered considerable attention from the research community in recent years. However, these systems are not ideally suited for situations where social distancing is crucial for people's safety, such as during the COVID-19 pandemic. In this paper, we study this problem through a use case scenario: recommending items for observation during an open-door hospital visit. We propose an approach for Side-LARS (Social-Distance preserving LARS), a trajectory and user-based collaborative filtering algorithm, that incorporates location data, user behaviors and social distancing constraints to provide personalized recommendations. The experimental results demonstrate the effectiveness of the proposal in maintaining social distancing while providing personalized recommendations.

Keywords: Location-Aware Recommender Systems · Social distance · Implicit ratings · Synthetic dataset generation

1 Introduction

In recent years, Location-Aware Recommender Systems (LARS) have gained attention for leveraging spatial features of users and items to generate relevant

recommendations. Users often prefer nearby items due to the convenience of access, making LARS a valuable extension of traditional recommender systems and a significant subset of Context-Aware Recommender Systems (CARS) [4].

The COVID-19 pandemic has underscored the importance of social distancing, particularly in public areas like hospitals. Traditional LARS are insufficient in scenarios where maintaining physical distance is critical. This paper introduces Side-LARS (*Social-Distance preserving LARS*), which addresses this gap by providing location-aware recommendations that also ensure social distancing. The proposed system offers personalized item routes, updating recommendations based on the user's location, item locations, and social distancing constraints. This approach is demonstrated through a hospital use case, where the system guides visitors via trajectories through items like informative posters and medical videos.

2 Related Work

Location-aware recommender systems (LARS) have become crucial in enhancing user experiences by incorporating geographical information, especially in sectors like travel, hospitality, and local services. They adapt dynamically to users' locations and needs, improving satisfaction by offering relevant items. Previous works [5, 13] provide comprehensive reviews of LARS, analyzing various approaches and applications. This section focuses on item trajectory recommendation approaches incorporating location information, user behaviors, and constraints for personalized recommendations in mobile environments.

Several studies leverage Location-Based Social Networks (LBSNs) where recommender systems use check-in data to identify interesting locations. For example, CLoSe [3] suggests POI sequences using Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) models. Another example is POIBERT [11], which combines LSTM and BERT to predict and recommend POI sequences, using DBSCAN for clustering regions of interest. Similarly, the approach in [14] uses X-means clustering and graph embedding to recommend POI itineraries based on users' travel patterns.

In addition to LBSNs, some works integrate real-time location data from GPS to recommend personalized item trajectories. For instance, POIBERT [7] is a BERT model adapted to recommend tour itineraries based on historical POI visit trajectories. BTRec [6] extends POIBERT by incorporating demographic information to improve prediction accuracy. GeoSAN [9], a Geography-aware Sequential Recommender based on a Self-Attention Network, uses geographic information to recommend the next POI based on user preferences and locations.

The COVID-19 pandemic has highlighted the need for recommender systems that consider social distancing and crowd-awareness. Side-CARS [8] introduces the idea of preserving social distance in recommendations, although it does not provide a specific solution. Recent works emphasize the importance of social distancing in route planning [2, 10], but there remains a gap in developing comprehensive solutions for integrating these constraints into LARS.

3 Location-Aware Recommender Architecture

The proposed architecture is structured around a centralized system that prioritizes server-based computation, partitioned into three distinct layers: the *View Layer*, where users engage with the items within the *Environment*; the *Logic Layer*, which forms the computational heart of the system; and the *Data Layer*, where all necessary data are stored and managed. At the core of the Logic Layer resides the Side-LARS algorithm, a sophisticated component of the *Recommendation Engine* that takes real-time user locations and social distancing into account, delivering personalized item trajectories. This system is designed to dynamically update and adapt its recommendations as users navigate through the space. We have defined an approach for Side-LARS based on the post-filtering paradigm [1]. It is inspired by the approach presented in [12], which was extended by incorporating an additional constraint for social distancing [8]. Beforehand, it is assumed that all users are making good use of the recommender system.

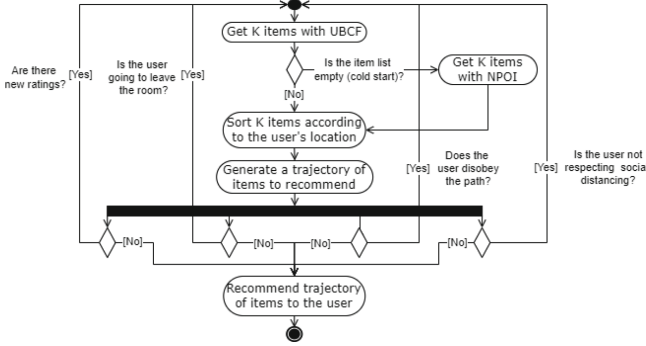


Fig. 1. Activity diagram presenting Side-LARS algorithm pipeline.

The algorithm begins by finding other users with similar preferences to the target user through *User-Based Collaborative Filtering (UBCF)*. Known ratings provided by these similar users are considered to estimate potential ratings that the target user might assign to different items. Based on these estimations, the top-k items (i.e., the k items with the best-predicted ratings not yet seen by the user) are identified as potential candidates for recommendation. If there are not enough data for UBCF to provide results (cold start problem), then the k -nearest points of interest are collected as candidate items, using the *Nearest Point of Interest (NPOI)* strategy. In the post-filtering phase, the resulting items are then reordered, if necessary, to minimize the distance the user will need to go through to access those k items. The shortest path passing through all those k items is computed. After that, several circumstances could trigger a reevaluation of the recommendation process, as shown in Fig. 1. For example, the social distancing constraint is applied to prevent overcrowding by enforcing a minimum Euclidean

distance threshold between the recommended item locations for different users, being initially verified before generating the trajectory for optimization purposes.

4 Experimental Evaluation

The digital transformation initiative of the Hospital San Juan de Dios in Zaragoza (Spain) takes a significant leap into the metaverse with the creation of an interactive virtual hospital environment (<https://virtual.hsjudzaragoza.es>), developed in collaboration with the Spanish company Imascono. This virtual scenario mirrors a real-world use case where an open-door visit to a hospital allows new residents or general visitors to familiarize themselves with the hospital environment, services, and facilities. During such events, visitors can explore the hospital freely. In our work, we exploit this virtual scenario to evaluate a proposal for Side-LARS that is suitable for its real equivalent, that is, a real hospital where physical distances among users should be kept and overcrowded areas need to be avoided.

For experimental evaluation, Python 3.8.16 and the recommendation algorithms of the Surprise (<https://surpriselib.com>) library were used. We exploited both real and synthetic datasets. We started by extracting relevant data from Imascono’s Open Search database (<https://opensearch.org>), which includes details about the virtual hospital environment. This data consists of objects (e.g., posters, screens), real-time user locations (geographic coordinates), and interactions with objects (e.g., click, close, play, pause). The dataset has 12 columns and 104,056 rows, covering several months from early 2022 to mid-2023, and includes information such as 19,483 unique user IDs, 27,767 object positions, and 93,237 recorded action timestamps.

Since this real dataset lacks explicit user interest in these objects, we use AUTO-DataGenCARS+, a synthetic data generator for recommender systems, to generate implicit ratings based on user behaviors and predefined rules related to the time spent interacting with objects. We extended AUTO-DataGenCARS+ to produce these implicit ratings, which in a real scenario would be captured using similar rules.

Figure 2 shows a heatmap of user locations in the hospital space using the mixed real and synthetic dataset. The heatmap reveals user spatial patterns and activity preferences over time, with yellow for low-traffic areas, pink for high-traffic areas, and an intermediate tone for moderate traffic. High traffic is concentrated around items near the hospital’s main entrance, indicated by blue dots. Intense red areas may suggest user dissatisfaction due to potential crowding, leading to unsafe distancing.

The Side-LARS approach addresses user crowding by incorporating social distancing constraints in the recommendation process. Initially, a user-based collaborative filtering approach with the KNNBaseline algorithm is employed to recommend $k = 5$ candidate items. The KNNBaseline algorithm was chosen for its superior performance, with precision (0.815), recall (0.9575), and F1-score

(0.864), compared to other KNN-based recommenders (KNNBasic, KNNWithMeans, and KNNWithZScore) using AUTO-DataGenCARS+. Finally, the k items recommended to users are reordered to minimize the traversed distance.

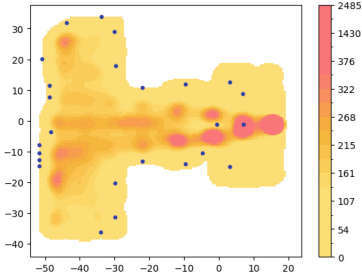


Fig. 2. Hospital visit heatmap.

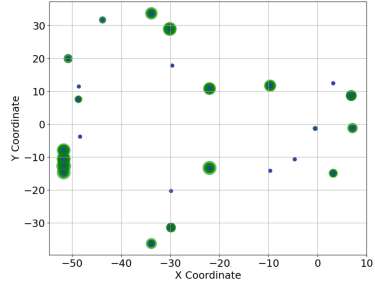


Fig. 3. User-item interactions.

A one-hour Side-LARS evaluation was conducted with 100 users from the mixed real and synthetic dataset. Figure 3 shows users scattered with low density per item, indicating that the Side-LARS algorithm helps mitigate social-distance infringements, which is expected to lead to a more pleasant user experience. Managing visitor distribution and maintaining personal distance is important to ensure the comfort of users, especially where social distancing is crucial.

5 Conclusions and Future Work

In this study, we addressed the challenge of combining location-based recommendations with social distancing requirements, particularly relevant in settings like hospitals. Our innovation, the Side-LARS algorithm, provides users with smart suggestions for maintaining safe distances. We tested Side-LARS in a realistic model of Hospital San Juan de Dios using both real and synthetic data. This approach helped us understand movement and interactions within the virtual hospital. Our experimental results highlight the potential benefits of our proposal to distribute users better in space, which, to our knowledge, is a novel contribution as no other work has proposed Side-CARS or Side-LARS.

For future work, we plan to explore recommendation approaches that incorporate additional context variables beyond location, transitioning from Side-LARS to more generic Side-CARS. We are also examining virus propagation models to evaluate the practical impact of our proposals more precisely. Additionally, we intend to conduct in-depth research in other domains, such as tourism, where our approach has promising applications.

Acknowledgments. This publication belongs to the following project: PID2020-113037RB-I00, funded by MICIU/AEI/10.13039/501100011033. Besides the NEAT-AMBIENCE project, we also acknowledge the Government of Aragon (COSMOS research group; last group reference: T64.23R). We thank the IMASCONO company for collaborating and providing the dataset related to the virtual space “Hospital San Juan de Dios”. This work has also been partially funded by the Department of Big Data and Cognitive Systems at the Technological Institute of Aragon, under Retech Tourism-Spain Living Lab Agreement within the “PLAN FOR RECOVERY, TRANSFORMATION AND RESILIENCE - FINANCED BY THE EUROPEAN UNION - NEXTGENERATIONEU” and by the Autonomous Community of Aragon.

References

1. Adomavicius, G., Tuzhilin, A.: Context-Aware Recommender Systems. Springer, Heidelberg (2011). <https://doi.org/10.1007/978-1-4899-7637-6.6>
2. Anastasiou, C., Costa, C., Chrysanthis, P.K., Shahabi, C., Zeinalipour-Yazti, D.: ASTRO: reducing COVID-19 exposure through contact prediction and avoidance. *ACM Trans. Spatial Algor. Syst. (TSAS)* **8**(2), 1–31 (2021)
3. Baral, R., Iyengar, S.S., Li, T., Balakrishnan, N.: CLOSe: contextualized location sequence recommender. In: 12th ACM Conference on Recommender Systems (RecSys), pp. 470–474. ACM (2018)
4. del Carmen Rodríguez-Hernández, M., Ilarri, S.: AI-based mobile context-aware recommender systems from an information management perspective: progress and directions. *Knowl.-Based Syst.* **215**, 1–29 (2021)
5. Haruna, K., Musa, A., Yunusa, Z., Ibrahim, Y., Jibia, F., Rabi, N.: Location-aware recommender system: a review of application domains and current developmental processes. *Sci. Inf. Technol. Lett.* **2**(1), 28–42 (2022)
6. Ho, N.L., Lee, R.K.W., Lim, K.H.: BTRec: BERT-based trajectory recommendation for personalized tours, pp. 1–9 (2023)
7. Ho, N.L., Lim, K.H.: POIBERT: a transformer-based model for the tour recommendation problem. In: 2022 IEEE International Conference on Big Data (Big Data), pp. 5925–5933. IEEE (2022)
8. Ilarri, S., Trillo-Lado, R., Delot, T.: Social-distance aware data management for mobile computing. In: 18th International Conference on Advances in Mobile Computing & Multimedia (MoMM), pp. 138–142. ACM (2021)
9. Lian, D., Wu, Y., Ge, Y., Xie, X., Chen, E.: Geography-aware sequential location recommendation. In: 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD), pp. 2009–2019. ACM (2020)
10. Liu, J., Wood, K.L., Lim, K.H.: Strategic and crowd-aware itinerary recommendation. In: Dong, Y., Mladenić, D., Saunders, C. (eds.) *Machine Learning and Knowledge Discovery in Databases: Applied Data Science Track (ECML PKDD)*. Lecture Notes in Computer Science (LNCS), vol. 12460, pp. 69–85. Springer, Heidelberg (2021). https://doi.org/10.1007/978-3-030-67667-4_5
11. Noorian, A., Harounabadi, A., Hazratifard, M.: A sequential neural recommendation system exploiting BERT and LSTM on social media posts. *Complex Intell. Syst.* 1–24 (2023)
12. Rodríguez Hernández, M.D.C., Ilarri Artigas, S., Trillo, R., Hermoso, R.: Context-aware recommendations using mobile P2P. In: 15th International Conference on Advances in Mobile Computing & Multimedia (MoMM), pp. 82–91. ACM (2017)

13. Sánchez, P., Bellogín, A.: Point-of-interest recommender systems based on location-based social networks: a survey from an experimental perspective. *ACM Comput. Surv.* **54**(11s), 1–37 (2022)
14. Zhou, J., Gu, Y., Lin, W.: Complementing travel itinerary recommendation using location-based social networks. In: *Fifth IEEE International Conference on Internet of People (IoP)*, pp. 1876–1881. IEEE (2019)

Author Index

A

Amagata, Daichi I-146, II-201
An, Jing I-194
Anyimadu, Eric Ababio I-231
Aritsugi, Masayoshi II-142

B

Bai, Yanbing I-194
Bakir-Gungor, Burcu II-69
Bang, L. K. II-3
Bao, Q. T. II-3
Bedo, Marcos I-75
Bernasconi, Anna II-79
Bianchini, Devis II-18
Billah, Hafsa II-125
Bogorny, Vania II-79
Boughanem, Mohand II-306

C

Caballero, Marcos I-267
Cai, Chuan I-163
Canahuate, Guadalupe I-231
Carvalho, Jonata Tyska II-79
Casanova, Marco A. I-93
Chakravarthy, Sharma II-125
Chang, Qiong II-248
Chen, Cheng-Wei I-249
Cheng, Dawei I-3
Coelho, Gustavo M. C. I-93
Cohen, Idan I-124

D

De Antonellis, Valeria II-18
De Grandis, Luca I-222
de Oliveira, Aiko R. I-93
de Oliveira, Daniel I-75
Decker, Stefan II-53
Ding, Zhiming I-108
Dkaki, Taoufiq II-306

E

El-Shaikh, Amir II-217

F

Faci, Noura II-320
Faiz, Rim II-306
Fei, Yongjian I-3
Feijó, Lucas I-93
Ferroudj, Wissam II-320
Fukuda, Keishi I-18
Fuller, Clifton David I-231
Funamoto, Naoya I-43

G

García, Grettel M. I-93
Garcia, Robinson L. S. I-93
Garda, Massimiliano II-18
Götz, Manuel II-149
Granata, Francesco Maria I-222
Gu, Jianhua I-153

H

Hammami, Eya II-306
Hang, Haitian II-185
Hara, Takahiro I-146, II-201
Hatano, Kenji I-258
Hemid, Ahmad II-53
Hermoso, Ramón I-267
Hieu, D. M. II-3
Hong, Hyejin I-258
Hou, Linlin II-272
Hu, Junjie I-194
Hua, Rui I-194
Huang, Yue I-3

I

Ide, Ichiro II-233
Ilarri, Sergio I-267
Inokuchi, Akihiro I-43
Izquierdo, Yenier T. I-93

J

- Jayakumar, Megha II-53
 Ji, Yuchen I-146, II-201
 Jia, Zhongfeng I-163

K

- Kato, Tatsuya II-233
 Kawabata, Tomoki II-116
 Kayem, Anne V. D. M. II-174
 Kha, H. N. II-3
 Kheddouci, Hamamache II-320
 Khetarpaul, Sonia I-215
 Khat, Abderrahmane II-53
 Khoa, D. T. II-3
 Kimura, Yusuke I-258
 Kiran, Rage Uday II-272
 Kobayashi, Suomi I-81
 Koga, Hisashi II-257
 Koh, Jia-Ling I-249
 Komamizu, Takahiro II-233

L

- Lange, Christoph II-53
 Lemos, Melissa I-93
 Li, Bingxin II-110
 Li, Jianhua I-34
 Li, Jiyi I-194
 Li, Jun I-207
 Li, Mingzhang II-272
 Li, Rui I-194
 Li, Yunxuan II-288
 Li, Zhao II-272
 Li, Zhishuai I-139
 Li, Ziyue I-139
 Lin, Xinrui II-37
 Liu, Chenxi I-139
 Liu, Jiamin I-163
 Liu, Jie I-179
 Liu, Lei I-163
 Liu, Wenxiang I-34
 Liu, Yishan II-159
 Loc, V. C. P. II-3
 Lv, Xinjie I-108

M

- Ma, Qiang I-18
 Ma, Shicong I-153
 Machado, Vanessa Lago II-79
 Mao, Hangyu I-139

- Marai, G. Elisabeta I-231
 Matsugu, Shohei I-81
 Meinel, Christoph II-174
 Melchiori, Michele II-18
 Mendonça, Israel II-142
 Mitschang, Bernhard II-288
 Miyazaki, Jun II-248

N

- Nagpal, Aryan I-215
 Nakano, Masashi II-248
 Narang, Aarush I-215
 Nascimento, Eduardo R. S. I-93
 Niazmand, Emetis I-59

O

- Oro, Ermelinda I-222

P

- Pan, Yuchen I-207
 Parada, Raúl I-267
 Peng, Tao I-207
 Peng, Yiqiang II-37
 Pinheiro, João P. I-93
 Portela, Tarlis Tortelli II-79

Q

- Qu, Muzi I-179
 Quix, Christoph II-53

R

- Reich, David II-174
 Renso, Chiara II-79
 Rodríguez-Hernández, María del Carmen
 I-267
 Rost, Maximilian II-149
 Rubio, Óscar J. I-267
 Ruffolo, Massimo I-222

S

- Santos, Lúcio F. D. I-75
 Sasaki, Yuya II-201
 Schirmeier, Frank II-149
 Schuiki, Laura II-288
 Seeger, Bernhard II-217
 Shahriar, Md Hasan II-174
 Sharma, Dolly I-215
 Shibayama, Akihiro II-142
 Shiokawa, Hiroaki I-81

Silva-Leite, João I-75
Sinha, Shreya I-215
Soisalon-Soininen, Eljas II-217
Stach, Christoph II-288
Su, Qiong I-139
Su, Wei I-163
Sudo, Ryuichi II-95
Sun, Jianling II-185

T

Takano, Taisei II-257
Takashima, Ikuto II-142
Takeuchi, Yukiko II-142
Tang, Xiu II-185
Toda, Hiroyuki II-95, II-116
Trillo-Lado, Raquel I-267
Trong, N. D. P. II-3
Trung, P. H. T. II-3

V

Vidal, Maria-Esther I-59
Voskergian, Daniel II-69

W

Wang, Jinyuan I-3
Wang, Shuai I-179
Wang, Xiaofei II-37
Wang, Yu II-159
Wang, Yunlan I-153
Wang, Zhihui II-110, II-159
Wang, Zhong I-3
Weber, Mauro I-75
Wilkner, Dennis II-149

X

Xiao, Yuxi I-194
Xie, Jinhao I-3

Xie, Wenjian I-163
Xing, Wenbiao II-159
Xu, Jianhua I-207
Xu, Zenghui II-272
Xue, Yulin I-207

Y

Yakhini, Zohar I-124
Yamada, Junya I-146
Yan, Jin I-108
Yang, Guang I-179
Yang, Jianwen I-108
Yang, Sun I-139
Yasuda, Kotaro II-142
Ye, Dan I-179
Ye, Lei I-34
Yehezkel, Aviv I-124
Yousef, Malik II-69
Yu, Ting II-272
Yu, Yong I-3
Yuan, Yongna I-163
Yue, Wenli I-163

Z

Zhang, Ji II-272
Zhang, Peng II-272
Zhang, Qihong I-108
Zhang, Xinhua I-231
Zhao, Hai I-3
Zhao, Rui I-139
Zhao, Tianhai I-153
Zheng, Yuhao I-34
Zhong, Hua I-179
Zhou, Bo II-185
Zhu, Meiling I-108
Zhu, Yan II-37
Zhu, Zeyang I-3