

Carlos Ordonez
Ladjel Bellatreche (Eds.)

LNCS 11031

Big Data Analytics and Knowledge Discovery

20th International Conference, DaWaK 2018
Regensburg, Germany, September 3–6, 2018
Proceedings



 Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, Lancaster, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Zurich, Switzerland

John C. Mitchell

Stanford University, Stanford, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

C. Pandu Rangan

Indian Institute of Technology Madras, Chennai, India

Bernhard Steffen

TU Dortmund University, Dortmund, Germany

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbrücken, Germany

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

Carlos Ordonez · Ladjel Bellatreche (Eds.)

Big Data Analytics and Knowledge Discovery

20th International Conference, DaWaK 2018
Regensburg, Germany, September 3–6, 2018
Proceedings

Editors

Carlos Ordonez
University of Houston
Houston, TX
USA

Ladjel Bellatreche
LIAS/ISAE-ENSMA
Chasseneuil-du-Poitou
France

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-98538-1 ISBN 978-3-319-98539-8 (eBook)
<https://doi.org/10.1007/978-3-319-98539-8>

Library of Congress Control Number: 2018950524

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

© Springer Nature Switzerland AG 2018

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

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

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

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

Preface

Big data analytics and knowledge discovery remain hot research areas for both academia and the software industry, further fueled by advances in hardware and software. Important research topics associated with these major themes include data lakes (schema-free repositories), database design (ER modeling, prototyping), data integration (especially linking structured and semistructured data sources), big data management (mixing relational tables, text and any files), query languages (SQL and beyond), scalable analytic algorithms, parallel systems (cloud, parallel database systems, Spark, MapReduce, HDFS), theoretical foundations, and practical applications.

With a track record of 20 editions, the International Conference on Big Data Analytics and Knowledge Discovery (DaWaK) has established itself as a high-quality forum for researchers, practitioners and developers in the field of big data analytics. This year's conference (DaWaK 2018) built on this tradition, facilitating the interdisciplinary exchange of ideas, theory, techniques, experiences, and future research directions. DaWaK 2018 aimed to introduce innovative principles, methods, models, algorithms, industrial products, and experiences to solve challenging problems faced in the development of new-generation data management and analytic systems in the big data era.

Our call for papers attracted 76 papers, from which the Program Committee finally selected 13 full papers and 16 short papers, yielding an acceptance rate of 17% for full papers and 38% overall. Each paper was reviewed by at least three reviewers and in some cases up to five. Accepted papers cover a number of broad research areas on both theoretical and practical aspects. Some trends found in accepted papers include new generations of data warehouses, data lakes, data pre-processing, data mining, cloud computing, query processing, sequences, graph analytics, privacy-preserving data mining, and parallel processing. On the other hand, the program featured interesting case studies on social networks, Twitter sentiment analysis, understanding ground transportation modes, and E-commerce, among others.

For this 20th edition of DaWaK, we were pleased to have Prof. Il-Yeol Song from Drexel University (USA) as keynote speaker, giving an intriguing talk entitled: "Smart Aging: Topics, Applications, Technologies, and Agenda." Il-Yeol is an ACM Distinguished Scientist, an ER Fellow, and recipient of the 2015 Peter P. Chen Award in Conceptual Modeling.

Thanks to the history and reputation of DaWaK, editors of well-known journals agreed to receive extended versions of best papers selected from our program. This year, we were pleased to have two special issues in: *Data and Knowledge Engineering* (DKE, Elsevier) and *Transactions on Large-Scale Data- and Knowledge-Centered Systems* (TLDKS, Springer).

We would like to thank all authors for submitting their papers to DaWaK 2018 and we hope they submit again in the future. We express our gratitude to all the Program Committee members who provided high-quality reviews. We appreciate the great

efforts of Amin Anjomshoaa for helping extend the confdriver system with several innovations to improve paper reviews, to help in deciding between full and short length, to manage a conference-to-journal long-term review process, and to create an interesting, packed, thought-provoking conference program. Finally, we would like to especially thank Gabriela Wagner for her endless help and patience.

For conference attendees, we hope they enjoyed the technical program, informal meetings, and interaction with colleagues from all over the world; and of course, we are confident they liked the picturesque city of Regensburg, Germany. For the readers of these proceedings, we hope these papers are interesting and they give you ideas for future research.

September 2018

Carlos Ordonez
Ladjel Bellatreche

Organization

Program Committee Co-chairs

Carlos Ordonez University of Houston, USA
Ladjel Bellatreche ISAE-ENSMA, France

Program Committee

Alberto Abelló Universitat Politècnica de Catalunya, Spain
Toshiyuki Amagasa University of Tsukuba, Japan
Elena Baralis Politecnico di Torino, Italy
Ladjel Bellatreche ENSMA, France
Sadok Ben Yahia Faculty of Sciences of Tunis, Tunisia
Jorge Bernardino ISEC - Polytechnic Institute of Coimbra, Portugal
Vasudha Bhatnagar Delhi University, India
Omar Boussaid University of Lyon/Lyon 2, France
Stephane Bressan National University of Singapore, Singapore
Wellington Cabrera Teradata, USA
Sharma Chakravarthy The University of Texas at Arlington, USA
Isabelle Comyn-Wattiau ESSEC Business School, Paris, France
Alfredo Cuzzocrea University of Trieste, Italy
Laurent d'Orazio University of Rennes 1, France
Soumyava Das Teradata, USA
Karen Davis Miami University, USA
Claudia Diamantini Università Politecnica delle Marche, Italy
Josep Domingo-Ferrer Universitat Rovira i Virgili, Spain
Dejing Dou University of Oregon, USA
Curtis Dyreson Utah State University, USA
Markus Endres University of Augsburg, Germany
Leonidas Fegaras The University of Texas at Arlington, USA
Filippo Furfaro DIMES, University of Calabria, Italy
Pedro Furtado Universidade de Coimbra, Portugal
Carlos Garcia-Alvarado Autonomic LLC, USA
Javier Garcia-Garcia C3 UNAM University, Mexico
Kazuo Goda The University of Tokyo, Japan
Matteo Golfarelli DISI, University of Bologna, Italy
Sergio Greco University of Calabria, Italy
Abdelkader Hameurlain Paul Sabatier University, France
Takahiro Hara Osaka University, Japan
Frank Hoppner Ostfalia University of Applied Sciences, Germany
Yoshiharu Ishikawa Nagoya University, Japan
Stéphane Jean LIAS/ISAE-ENSMA and University of Poitiers, France

Lili Jiang	Umeå University, Sweden
Selma Khouri	LCSI/ESI (Algeria) and LIAS/ISAE-ENSMA (France), Algeria
Uday Kiran Rage	University of Tokyo, Japan
Nhan Le-Thanh	Nice Sophia Antipolis University, France
Jens Lechtenboerger	Westfälische Wilhelms–Universität Münster, Germany
Carson Leung	University of Manitoba, Canada
Sofian Maabout	University of Bordeaux, France
Yannis Manolopoulos	Aristotle University of Thessaloniki, Greece
Patrick Marcel	Université François Rabelais Tours, France
Jun Miyazaki	Tokyo Institute of Technology, Japan
Anirban Mondal	Ashoka University, India
Yasuhiko Morimoto	Hiroshima University, Japan
Makoto Onizuka	Osaka University, Japan
Carlos Ordonez	University of Houston, USA
Alex Poulouvassilis	Birkbeck, University of London, UK
Praveen Rao	University of Missouri-Kansas City, USA
Goce Ristanoski	Data61, CSIRO, Australia
Oscar Romero	Universitat Politècnica de Catalunya, Spain
Laura Rusu	IBM, Australia
Ilya Safro	Clemson University, USA
Sherif Sakr	Software Systems Research Group, NICTA, University of New South Wales, Australia
Abhishek Santra	University of Texas at Arlington, USA
Alkis Simitsis	HP Labs, USA
Emanuele Storti	Università Politecnica delle Marche, Italy
Aditya Telang	University of Texas at Arlington, USA
Olivier Teste	IRIT, University of Toulouse, France
Dimetri Theodoratos	New Jersey Institute of Technology, USA
Predrag Tomic	Washington State University, USA
Panos Vassiliadis	University of Ioannina, Greece
Robert Wrembel	Poznan University of Technology, Poland
Yinghui Wu	Washington State University, USA
Wai GenYee	Grubhub, USA
Yinuo Zhang	Teradata, USA
Yiqun Zhang	VoltDB Inc., USA

Additional Reviewers

Mohammed El Malki	Toulouse University/IRIT-UMR5505, France
Sergio Martínez	Universitat Rovira i Virgili, Catalonia
Luis del Vasto	Universitat Rovira i Virgili, Catalonia
Michael Bamiloshin	Universitat Rovira i Virgili, Catalonia
Calvin S. H. Hoi	University of Manitoba, Canada
Fan Jiang	University of Northern British Columbia, Canada
Adam G. M. Pazdor	University of Manitoba, Canada

Oluwafemi A. Sarumi	Federal University of Technology – Akure, Nigeria
Neelabh Pant	Walmart, Plano, Texas, USA
Abhishek Santra	University of Arlington, Texas, USA
Soumyava Das	Teradata, CA, USA
Aggeliki Dimitriou	National Technical University of Athens, Greece
Michael Lan	New Jersey Institute of Technology, USA
Xiaoying Wu	Wuhan University, China
Daniele Apiletti	Politecnico di Torino, Italy
Andrea Pasini	Politecnico di Torino, Italy
Eliana Pastor	Politecnico di Torino, Italy
Anastasios Gounaris	Aristotle University of Thessaloniki, Greece
Irina Trubitsyna	University of Calabria, Italy
Gianvincenzo Alfano	University of Calabria, Italy
Francesco Parisi	University of Calabria, Italy
Sharanjit Kaur	University of Delhi, India
Daniel Ernesto Lopez Barron	University of Missouri-Kansas City, USA
Amira Mouakher	Faculty of Sciences of Tunis, Tunisia

Smart Aging: Topics, Applications, Technologies, and Agenda (Abstract of Keynote Speaker)

Il-Yeol Song

College of Computing and Informatics, Drexel University,
Philadelphia, PA, USA
songiy@drexel.edu

Abstract. Aging is a rapidly growing social problem in the developed world. It is critically important to mitigate the effects of aging, improve elderly people's life, and improve overall quality of healthcare environments. Smart aging addresses those challenges by intelligently utilizing modern biomedical, digital healthcare, big data computing & analytics, IOT, and communication technologies. In this talk, I will first review several innovative R&D projects and services for smart aging. I will then present a comprehensive review of various research activities on smart aging from the content analysis of public web pages and the web pages of NIH funded research projects related to smart aging. I will then cover recent developments in big data technologies for smart health, including healthcare data warehouses, data lakes and big data analytics. I will conclude my talk with a summary of suggestions on smart aging projects and research topics in smart aging.

Keywords: Smart aging · Big data technologies · Smart health
Healthcare data warehouses · Healthcare data lake

Contents

Graph Analytics

Graph BI & Analytics: Current State and Future Challenges.	3
<i>Amine Ghrab, Oscar Romero, Salim Jouili, and Sabri Skhiri</i>	
Community Detection in Who-calls-Whom Social Networks.	19
<i>Ciprian-Octavian Truică, Olivera Novović, Sanja Brdar, and Apostolos N. Papadopoulos</i>	
FedS: Towards Traversing Federated RDF Graphs	34
<i>Qaiser Mehmood, Alok Kumar Jha, Dietrich Rebolz-Schuhmann, and Ratnesh Sahay</i>	

Case Studies

Adversarial Spiral Learning Approach to Strain Analysis for Bridge Damage Detection.	49
<i>Takaya Kawakatsu, Akira Kinoshita, Kenro Aihara, Atsuhiko Takasu, and Jun Adachi</i>	
CoRe: Generating a Computationally Representative Road Skeleton - Integrating AADT with Road Structure	59
<i>Rohith Reddy Sankepally and K. S. Rajan</i>	
E-Commerce Product Recommendation Using Historical Purchases and Clickstream Data.	70
<i>Ying Xiao and C. I. Ezeife</i>	
Effective Classification of Ground Transportation Modes for Urban Data Mining in Smart Cities.	83
<i>Carson K. Leung, Peter Braun, and Adam G. M. Pazdor</i>	
Location Prediction Using Sentiments of Twitter Users	98
<i>Ritu Singh and Durga Toshniwal</i>	

Classification and Clustering

A Clustering Model for Uncertain Preferences Based on Belief Functions . . .	111
<i>Yiru Zhang, Tassadit Bouadi, and Arnaud Martin</i>	
A Novel Committee-Based Clustering Method	126
<i>Sonia Fiol-Gonzalez, Cassio Almeida, Simone Barbosa, and Hélio Lopes</i>	

KMN - Removing Noise from K-Means Clustering Results	137
<i>Benjamin Schelling and Claudia Plant</i>	
Subset Labeled LDA: A Topic Model for Extreme Multi-label Classification	152
<i>Yannis Papanikolaou and Grigorios Tsoumakas</i>	
Third Party Data Clustering Over Encrypted Data Without Data Owner Participation: Introducing the Encrypted Distance Matrix	163
<i>Nawal Almutairi, Frans Coenen, and Keith Dures</i>	
Pre-processing	
An Efficient Prototype Selection Algorithm Based on Spatial Abstraction . . .	177
<i>Joel Luis Carbonera and Mara Abel</i>	
Web Usage Data Cleaning: A Rule-Based Approach for Weblog Data Cleaning.	193
<i>Amine Ganibardi and Chérif Arab Ali</i>	
Anonymization of Multiple and Personalized Sensitive Attributes	204
<i>Jerry Chun-Wei Lin, Qiankun Liu, Philippe Fournier-Viger, Youcef Djenouri, and Ji Zhang</i>	
TRANS-AM: Discovery Method of Optimal Input Vectors Corresponding to Objective Variables	216
<i>Hiroaki Tanaka, Yu Suzuki, Koichiro Yoshino, and Satoshi Nakamura</i>	
Sequences	
Discovering Periodic Patterns Common to Multiple Sequences	231
<i>Philippe Fournier-Viger, Zhitian Li, Jerry Chun-Wei Lin, Rage Uday Kiran, and Hamido Fujita</i>	
Discovering Tight Space-Time Sequences	247
<i>Riccardo Campisano, Heraldito Borges, Fabio Porto, Fabio Perosi, Esther Pacitti, Florent Masseglia, and Eduardo Ogasawara</i>	
Cloud and Database Systems	
CloudDBGuard: Enabling Sorting and Searching on Encrypted Data in NoSQL Cloud Databases	261
<i>Tim Waage and Lena Wiese</i>	
Query Processing on Large Graphs: Scalability Through Partitioning.	271
<i>Jay Bodra, Soumyava Das, Abhishek Santra, and Sharma Chakravarthy</i>	

Querying Heterogeneous Data in Graph-Oriented NoSQL Systems 289
*Mohammed El Malki, Hamdi Ben Hamadou, Max Chevalier,
 André Péninou, and Olivier Teste*

Selection of Bitmap Join Index: Approach Based
 on Minimal Transversals 302
Issam Ghabry, Sadok Ben Yahia, and M. Nidhal Jelassi

Scalable Random Sampling K-Prototypes Using Spark 317
*Mohamed Aymen Ben HajKacem, Chiheb-Eddine Ben N'cir,
 and Nadia Essoussi*

Data Mining

ERAPN, an Algorithm for Extraction Positive and Negative Association
 Rules in Big Data 329
Parfait Bemarisika and André Totohasina

Multistep-ahead Prediction: A Comparison of Analytical
 and Algorithmic Approaches 345
Fouad Bahrpeyma, Mark Roantree, and Andrew McCarren

Novel Data Segmentation Techniques for Efficient Discovery of Correlated
 Patterns Using Parallel Algorithms 355
*Amulya Kotni, R. Uday Kiran, Masashi Toyoda, P. Krishna Reddy,
 and Masaru Kitsuregawa*

Time Series Distance Density Cluster with Statistical Preprocessing 371
Ruizhe Ma, Soukaina Filali Boubrahimi, and Rafal Angryk

Debate Stance Classification Using Word Embeddings. 382
*Anand Konjengbam, Subrata Ghosh, Nagendra Kumar,
 and Manish Singh*

Author Index 397

Graph Analytics



Graph BI & Analytics: Current State and Future Challenges

Amine Ghrab^{1,2}(✉), Oscar Romero², Salim Jouili¹, and Sabri Skhiri¹

¹ EURA NOVA R&D, Mont-Saint-Guibert, Belgium
{[amine.ghrab](mailto:amine.ghrab@euranova.eu), [salim.jouili](mailto:salim.jouili@euranova.eu), [sabri.skhiri](mailto:sabri.skhiri@euranova.eu)}@euranova.eu

² Universitat Politècnica de Catalunya, Barcelona, Spain
oromero@essi.upc.edu

Abstract. In an increasingly competitive market, making well-informed decisions requires the analysis of a wide range of heterogeneous, large and complex data. This paper focuses on the emerging field of graph warehousing. Graphs are widespread structures that yield a great expressive power. They are used for modeling highly complex and interconnected domains, and efficiently solving emerging big data application. This paper presents the current status and open challenges of graph BI and analytics, and motivates the need for new warehousing frameworks aware of the topological nature of graphs. We survey the topics of graph modeling, management, processing and analysis in graph warehouses. Then we conclude by discussing future research directions and positioning them within a unified architecture of a graph BI and analytics framework.

1 Introduction

Graphs are fundamental and widespread structures that provide an intuitive abstraction for the modeling and analysis of complex, heterogeneous and highly interconnected data. They have the benefit of revealing valuable insights from content-based and topological properties of data. The great expressive power of graphs, along with their solid mathematical background, encourages their use for modeling domains having complex structural relationships. In the context of Big Data, the focus of organizations is often on handling the rising volume of their data. However the variety and complexity of data through the different phases of data capturing, modeling and analysis is at least equally important. The variety challenge is the most critical challenge in big data nowadays, and efficiently handling the variety of data sources is considered to be the main driver of success for data-driven organizations [1]. Graphs meet the requirements to be the perfect canonical data model for data integration systems [2] given (1) their capability to deal with semantic relativism and semantic heterogeneities, (2) they are semantically richer at least as any other model (so they can represent any semantics), (3) they allow to create multiple views from the same source, (4) and most importantly, graphs are extremely flexible to compose new graphs.

That is given two graphs, with one single edge a new graph could be directly created without affecting the existing ones. Therefore, graphs are suitable to deal with big data variety better than any other data model.

Furthermore, in industry, graph analysis is considered as *“possibly the single most effective competitive differentiator for organizations pursuing data-driven operations and decisions after the design of data capture”* according to Gartner, Inc., a research and advisory firm [3]. Indeed, large complex graphs have emerged in various fields and graph analytics are being increasingly used to solve complex real-world problems. In the financial sector for example, several types of fraud could be detected and prevented in auction and transaction networks [4]. In [5], the authors used bank transaction to build a financial transactions network, where each node represents a client, and each edge represents a transaction. As fraudsters tend to collaborate to orchestrate complex fraud at large scale, the probability that a customer is involved in a fraud depends on his neighborhood in the transaction graph. Graph analytics could be used to define and retrieve complex fraud patterns, or to score customers by fraud exposure. In [6], the authors built a Call Detail Record graph to understand the call routines and interactions between customers. This information can later be used to prepare marketing campaigns or to prevent customer churn.

It is clear that the topological properties of graphs are of big potential to decision-making systems. They supply these systems with a new class of complex structural business facts and measures that could be explored for making more accurate decision in data-driven organizations. In current information systems, Business Intelligence (BI) systems are critical for strategic decision making. Graph BI in particular, is emerging as the BI field that extends current BI systems with graph analytics capabilities. It enables graph-based insights such as detection of popular users or communities in social networks, or revealing hidden interaction patterns in financial networks. Graph BI can help address the above-mentioned big data applications since (1) data is interconnected in complex ways, but graphs can help reduce this complexity with intuitive data models and queries, (2) the data size is large, but data warehouses and Online Analytical Processing (OLAP) analysis are suitable for storage, organization, synthesis and analysis of large volumes of data, and (3) graph mining extends traditional techniques by including discovery of the topological properties, thus characterizing more precisely business applications. Traditional BI systems, and particularly data warehouses, were designed to support relational data management and analysis. Due to the fundamental difference between graph and relational data, the existing systems are not suitable for efficient graph analysis. The structure-driven management and analytics of graph data call for the development of novel data models, query processing paradigms and storage techniques. Therefore, as motivated by multiple research lines [7, 8], current BI and analytics systems need to be extended to efficiently support warehousing [9], processing [10], mining [11] and OLAP analysis [12, 13] of the graph structural and content-based information.

Figure 1 provides an overview of the different components of the envisioned graph BI system. While adopting a similar template as the traditional BI systems (i.e., it preserves the familiar data analytics workflow), graph BI extends current systems with graph-aware components that deliver graph-derived insights.

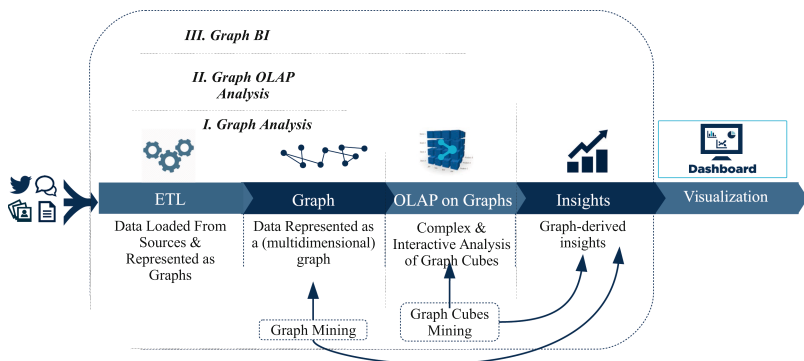


Fig. 1. Integration of Graph and Traditional BI

Note that through this paper the terms “graph and network”, “node and vertex”, and “edge and relationship” are often used interchangeably. The remainder of this paper presents the current state and the open challenges of graph BI & analytics, with a focus on graph warehousing. Section 2 discusses the topic of graph data modeling and management. Section 3 surveys the existing frameworks for graph analytics. Section 4 identifies future research directions and position them within a unified architecture of a graph BI & analytics framework.

2 Graph Data Modeling

The variety of data structures urges the need for equipping analysts with modeling and querying tools that are aware of the specific nature of each data model. The relational model and its implementations have been developed and matured for decades. However, they were pushed to their limits as the one-size-fits-all data management solutions. The wide adoption of the emerging NoSQL solutions by industry, while not yet as mature as relational model, proved the need to push databases into fields beyond the traditional business applications. More specifically, organizations experience an urgent need for models and techniques for efficient management of graph data. Indeed, graph models have the ability to deal with semantic relativism and heterogeneities, offer the flexibility to combine graphs, and support the capability to associate data and metadata.

2.1 Graph Models

According to the literature, the two main families of graphs are property graphs and knowledge graphs:

Property Graphs: Property graphs describe a directed, labeled and attributed multi-graph. Each real-world entity is represented by a node that contains its label and properties. The label denotes the “type” of the node (i.e., the class to which it belongs). Relationships between entities are represented using edges. The flexibility of property graph models allows the representation of rich structural properties, such as hierarchies and assertions. Property graphs were introduced in the database community to store schemaless data (due to their flexibility to absorb any semantics and attach data with metadata). In the literature, multiple query languages were designed to enable graph-oriented querying of property graphs [14]. However, there is no standard query language for property graphs. Therefore, graph database vendors defined their own graph traversal and query languages, such as Cypher and Gremlin. Cypher is an SQL-like declarative language, that uses isomorphism-based no-repeated-edges bag semantics. It was introduced by Neo4j, and is centered around pattern matching enriched by built-in algorithmic operators. Gremlin is a graph traversal language, built using Groovy, introduced by Apache TinkerPop3, that uses the homomorphism-based bag semantics.

Knowledge Graphs: The basic formalism behind knowledge graphs, used to describe and link resources, is the Resource Description Framework language (RDF), a W3C recommendation. The basic RDF block is the triple, a binary relationship between a subject and an object; i.e., $\langle \textit{subject}, \textit{predicate}, \textit{object} \rangle$. The subject and the predicate must be resources (i.e., identified by a URI), whereas the object can be either a resource or a literal (i.e., a constant value such a string or an integer). A set of RDF triples form an RDF graph. RDF Schema (RDFS), a W3C recommendation, was introduced to express basic constraints on RDF triples. In the same line, the Ontology Web Language (OWL) allows to express richer constraints and semantics. As OWL is serialized on top of RDFS, it results in a graph too. Importantly, it is also usual to refer to knowledge graphs (i.e., RDF(S) or OWL) as ontologies. This is because they can be translated to a fragment of First Order Logic (FOL), typically, within the family of Description Logics (e.g., in the case of OWL and OWL fragments), and benefit from generic reasoning algorithms in that field. Finally, the W3C recommendation to query knowledge graphs is the SPARQL Protocol and RDF Query Language (SPARQL). Relevantly, SPARQL enables to activate generic reasoning capabilities when querying knowledge graphs to infer non-explicit knowledge. Knowledge graphs were born within the semantic web stack and therefore initially thought for enabling interoperability and reasoning on semantic data. They are tightly related to the knowledge representation community and therefore thought to represent generic knowledge rather than data as for databases. For this reason, knowledge graph databases are referred to as triplestores rather than graph

databases. One key aspect with regard to property graphs is that they provide means (i.e., URI) to universally identify graph vertices and edges from external sources. This facilitates linking and sharing of data and metadata. However, unlike property graphs and traditional graph databases, which are optimized for graph traversal, they are primarily optimized for handling RDF triples. Another difference is that in property graphs properties could be directly added to edges as well as vertices. In essence, however, knowledge graphs are also graphs and can benefit from the traditional graph algebras presented in the database field.

2.2 Graph Management

Orthogonal to the previous classification there are two main approaches widely used for graph data management (regardless of property or knowledge graphs) at the logical/physical level. The first consists on the use of native graph data models and database engines. The second leverages alternative models, mainly the relational model. For the latter, the graph data is represented by a set of tables, i.e., node tables and edge tables. Traditionally, relational-based graph database engines have been related to triplestores and knowledge graphs, whereas native graph database engines were related to property graphs. Nevertheless, this is nowadays changing and it is currently possible to find native databases for knowledge graphs. These approaches are discussed next:

Native Graph Data Models: In recent years, the trend in developing graph data management systems has shifted to the development of native, relationship-oriented graph databases. Most native graph databases implement the property graph model or a variation of it. The problem of impedance mismatch is resolved since relationships are first-class citizen, and the data is represented as it is perceived without the need to project it on an intermediate representation. The data model is more straightforward to design, and the queries are more intuitive to formulate [15]. From a performance perspective, graph databases are optimized for graph traversals. The cost of traversing an edge is constant, and the overall cost of arbitrary navigation through the graph is much lower than the equivalent joins in relational tables. Subsequent implementation aspects such as graph query processing, indexing, storage, matching and retrieval which are specifically developed and tuned for graph workloads lead to better performances, especially for queries requiring multiple joins, or containing cycles or other complex patterns. However, they perform worse than the relational-based engines for analytical queries that perform scans over the whole graph. In the software market, established BI vendors are aware of the potential of native graph solutions and have already developed many graph databases such as Neo4j, DataStax Enterprise Graph, Oracle Spatial and Graph, Microsoft GraphEngine, IBM Graph, and Amazon Neptune.

Relational-Based Database Models: This approach benefits from the well-established relational model features, and enables a smooth integration with a

wide range of relational platforms. However, the relational model and its implementations fall short of meeting the requirements for (1) intuitive data modeling, (2) topology-aware graph querying (such as path retrieval and comparison, and graph pattern matching), and (3) traversal-optimized performances. Mapping graph data to relational representation raises the problem of impedance mismatch at the modeling and querying levels. For example, due to the fundamental difference between the two models, transformation of graph data to the relational model is a manual, complicated process, with a high risk of information loss during the transformation process. The relational model was designed to handle data such as sets of records and transactions instead of entities and connections. The relational query languages and query processing engines are optimized to perform table scans instead of traversals. Graph traversal is often simulated using expensive join operations, which incurs a heavy workload especially for highly interconnected tables. Moreover, the SQL is not suited to target the topology of the graph with queries such as pattern matching, neighborhood or path retrieval.

3 Graph Analytics

A plethora of graph analysis techniques were proposed in the literature to reveal interesting properties about the graph topology and the connectivity between graph elements. The core analysis operations of graphs are (1) graph traversal to assess reachability, find shortest paths, and retrieve the neighborhood, (2) metrics computation of local (e.g., centrality), and global properties (e.g., diameter), and (3) graph matching, such as subgraph isomorphism and pattern matching [14]. Most of these operations are supported by graph database engines. This section describes more advanced graph analytics and focuses on three pillars of graph analytics: graph OLAP, graph mining, and graph processing.

3.1 OLAP on Graphs

The multidimensional model is widely used to represent data in the warehouse. The business facts are stored following the multidimensional model in cubes that embed aggregated data denoted as measures, which are the metrics for the analysis. The measures are placed into the so-called multidimensional space, where dimensions are the factors influencing the values of the measures. OLAP techniques are widely used by BI analysts to conduct interactive and complex querying over large volume of data, from different perspectives and through different hierarchical levels, highlighting the items of interest, and then drilling down to the underlying data from which facts were inferred. The main approaches for the multidimensional design and OLAP analysis of cubes on graphs are:

Graph OLAP was among the first attempts to design a conceptual framework for OLAP analysis over a collection of homogeneous graphs [12]. Each graph of the collection is considered as a snapshot. Attributes are considered as the dimensions, and could be either attached to the whole graph snapshot, or to

single nodes. In the first case, the attributes of the snapshots are called informational dimensions. The aggregations of the graph are performed by overlaying a collection of graph snapshots and merging those with shared informational values. The analysis is referred to as informational OLAP aggregations, and consists in edge-centric snapshot overlaying. Thus only edges are merged and changed, with no changes made to the nodes. In the second case, the attributes of the nodes are called topological dimensions. Topological OLAP aggregations consist on merging nodes and edges by navigating through the nodes hierarchy. Qu et al. introduced a more detailed framework for topological OLAP analysis of graphs [16]. The authors discussed the structural aggregation of the graph following the OLAP paradigm. They presented techniques based on the properties of the graph measures for optimizing measure computations through the different aggregation levels. Berlingerio et al. [17] defined a multidimensional model similar to Graph OLAP, but where the dimensions are the labels of the edges, and presented a set of analytical graph-based measures.

Graph Cube is a framework for multidimensional analysis and cube computation over the different levels of aggregations of a graph [18]. It targets single, homogeneous, node-attributed graphs. A subset of the attributes of the nodes is considered as the dimensions. Following these so-called dimensions, the graph cube is obtained by restructuring the initial graph in all possible aggregation. Given n dimensional attributes, the framework introduces the cuboid query, which generates 2^n aggregate graphs (called graph cuboids). Crossboid is a second query introduced by Graph Cube to analyze the interrelationships between different graph cuboids. Pagrol is a Map-Reduce framework for distributed OLAP analysis of homogeneous attributed graphs [19]. Pagrol introduced the notion of Hyper Graph Cubes that extends the model of Graph Cube by in addition considering the attributes of the edges as dimensions, and introduced various optimization techniques for cubes computation and materialization. Ghrab et al. [20] extended those models with a framework for building OLAP cubes on heterogeneous attributed graphs. They presented an extension of property graphs tailored for multidimensional analysis and supporting dimension hierarchies.

Graph OLAP on RDF There is an active research line to generate OLAP cubes on top of RDF and RDF(S) graphs. Nebot [21] and Kämpgen [22] were two of the main attempts to bridge both areas. The former proposes a semi-automatic method for on-demand extraction of semantic data into an MD database. In this way, data could be analyzed using traditional OLAP techniques. The latter study the extraction of statistical data published using the QB vocabulary, a W3C standard, into an MD database. Both approaches moved the semantic data to a traditional data warehouse. Subsequent attempts avoided such approach and query graph data in an OLAP manner without moving it. For example, Beheshti et al., introduced a distributed framework for OLAP on RDF data [23]. They proposed GOLAP, a graph model for OLAP on graphs, and FSPARQL an extension to SPARQL for OLAP querying of RDF data. GOLAP introduced a rule-based approach for defining new dimensions on the graph. However, it was

not until the definition of the QB4OLAP vocabulary that cubes on RDF graphs could not be guaranteed to be MD-compliant. In [24], Varga et al. discusses the drawbacks of previous vocabularies, such as QB, to properly represent MD data and how QB4OLAP overcomes them. This way, resulting cubes can be properly analyzed with traditional OLAP algebras. Relevantly, Nath et al. present a framework to conduct ETL transformations on top of graph data to produce QB4OLAP-based cubes [25].

3.2 Graph Mining

Data mining refers to the process of discovering patterns or models for data. In contrast to querying that retrieves known patterns, mining enable the discovery of previously unknown, implicit information and knowledge embedded within a dataset. Traditionally, data mining techniques process data as collection of independent instances (i.e. observations). However, the recent emergence of graph structure as a rich data model involves a paradigm shift on how data mining algorithms can be applied. In fact, graph mining algorithms provide a new way of extracting and discovering latent insight from graphs by leveraging the relationships between entities. However, graph mining algorithms face three main challenges: (1) adapting the mining algorithms to make them graph-aware, (2) redesigning the algorithms to be implemented by those new high-performance techniques, and (3) storing and exploiting multiple but related graphs that serve for the same business purpose as in the graph warehouse.

A plethora of graph mining techniques were proposed in the literature such as graph clustering, frequent subgraph mining, proximity pattern mining and link prediction. These techniques are relevant in the BI context as they reveal interesting properties about the topology and the connectivity between business entities. For example, consider the case of recommender systems in e-commerce [26]. The domain could be represented as a bipartite purchase graph, with two types of nodes representing products and customers. An edge is added between a product and a customer if the latter has bought the product. Using graph mining such as graph clustering, two other graphs could be derived: (1) client similarity graph, and (2) product similarity graph. Mining these three graphs enables advanced analysis scenarios such as (1) customer profiling by detecting customer groups using community detection, and the leaders within each group using centrality, (2) product segmentation by detecting products representative of each segment, and (3) using link prediction, targeted marketing personalized to the customer's profile and tailored by current product trends.

The historical and integrated view provided by the data warehouse makes it a suitable backbone for offering a variety of analysis scenarios. In the graph warehouse context, graph mining could be combined with OLAP to offer more capabilities both during the phase of the design and also the analysis of the graph warehouse data. During the design phase, graph mining algorithms could be used to enrich the OLAP cubes with new types of topological dimensions and measures (e.g., PageRank, community). During the analysis, graph mining

could assist the analyst in complex tasks such as building summarized business-oriented views of the graph, providing new perspectives to analyze the graph, or discovering interesting or anomalous patterns within the large graph cube space. In the context of outlier detection, graphs provide an elegant framework to predict and describe outliers. For example, in the context of graph cubes mining, [27] developed a measure of interestingness of patterns in a graph cube, while [28] proposed an entropy-based filter to detect interesting associations between attributed nodes in a graph cube.

3.3 Graph Processing

To deal with large and evolving graphs, which is the case in data warehouses, graph BI systems need to integrate large-scale graph processing frameworks. Graph processing frameworks are designed to natively support the graph topology, and they offer graph programming models and abstractions to easily implement a multitude of graph algorithms. These frameworks have the capabilities to efficiently perform large scale, ad-hoc, and distributed computations over large graph data that exceed a single machine capacity. They offer features such as automatic graph partitioning, load balancing, network and disk transfer optimization, and fail-over of the processing tasks.

However, distributed graph processing poses additional challenges to centralized or traditional parallel data processing in that [29]: (1) graph structure is irregular which poses challenges to the graph data partitioning and limits parallelism, (2) computation is driven by the structure, which causes a poor memory locality and poses data transfer issues, and (3) algorithms traverse the partitioned graph in an exploratory way, and are iterative by nature, which is I/O intensive. To tackle these challenges and enable efficient large-scale graph analytics, different processing paradigms were introduced [30]:

- Hadoop Family frameworks: MapReduce denotes a programming model for large-scale data processing. Hadoop is an open-source framework that supports data-intensive distributed applications and clones the Google’s MapReduce framework. It is designed to process large amount of unstructured and complex data and runs on shared-nothing architectures. MapReduce frameworks are useful for content-based aggregation of graphs (e.g., graph cube aggregation), but they are not efficient for graph-specific computations [31].
- Synchronous frameworks: Pregel [32], and its open source implementation Apache Giraph, are distributed fault-tolerant graph processing frameworks designed to execute vertex-centric graph algorithms following the Bulk Synchronous Parallel processing (BSP) paradigm. BSP is a shared-nothing processing paradigm for parallel algorithms execution. The computation is done as a series of super-steps over a set of processing units, each having its local memory. Each super-step consists of three phases, first (1) each processing unit performs concurrently and locally its computations, then (2) data is exchanged between the different processes, finally (3) when a process finishes the computation and communication, it reaches the synchronization barrier

and it waits for the rest of processes to finish before proceeding to the next super-step. The advantage of this paradigm is that it ensures a deadlock-free computation. However, the downside is the execution time, where the system has to wait for the slowest machines to finish before proceeding.

- Asynchronous frameworks: In contrast to the synchronous shared-nothing processing frameworks, GraphLab [33] and PowerGraph [34] are asynchronous and follows the Gather-Apply-Scatter computational model, with shared memory abstraction. These frameworks might provide better performances, but incur more complexity and higher scheduling and consistency costs.
- Hybrid Systems: These frameworks enable a mixed workload of graph-parallel and data-parallel processing. GraphX [35] is a component of Apache Spark [36] developed for graph processing. It is a fault-tolerant, distributed, in-memory graph processing framework built on top of the Resilient Distributed Dataset abstraction. GraphX provides a set of primitive operators to load and interactively query the graph data. GRADOOP is a distributed framework for graph management and querying [37]. It introduces a new graph model that extends property graphs, support Cypher queries, and the queries are processed using of Apache Flink [38].

4 Future Research Directions

This paper calls for the development of novel intelligent, efficient and industry-grade graph warehousing systems. The potential directions include (1) further research on solving complex graph problems (e.g., subgraph isomorphism and graph partitioning), (2) building native graph components (e.g., native graph ETL operations, graph OLAP engines, and a multidimensional query language for graphs), and (3) the integration of artificial intelligence techniques to enable self-service BI for business users. To this end, machine learning should be integrated within the BI systems to automate the warehousing tasks from data preparation, to complex modeling and augmented analytics of graphs (e.g., automated discovery of multidimensional concepts, detection of interesting patterns, and forecasting of business KPIs evolution).

The modules missing for developing an industry-grade graph BI and analytics system are unified in the envisioned architecture presented in Fig. 2, and they summarize the future research directions as follows:

- Graph Extraction (1): This module allows the extraction of graph data from different data sources that could initially be in various formats and flowing at various rates such as graph streams. The data is cleaned to only capture entities that satisfy the quality constraints (e.g., contains the required attributes with valid values), which guarantees the reliability of data.
- Graph Construction & Enrichment (2): The captured graph data is integrated and formatted according to a given graph model. A promising research direction is the development of graph-aware ETL processes with native graph manipulation operations augmented with machine learning capabilities.

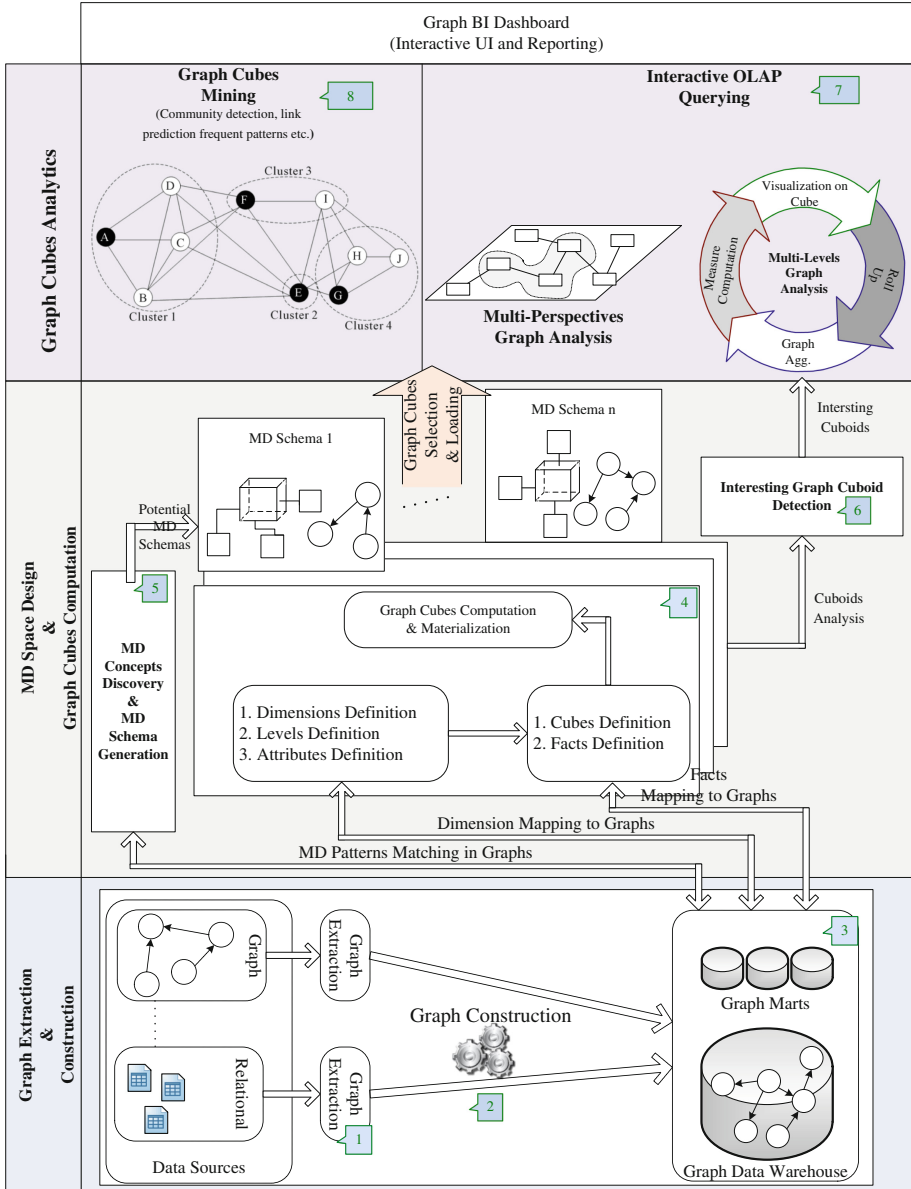


Fig. 2. Architecture of a Graph BI & analytics framework

ML techniques such as Information Retrieval and Automatic Natural Language Processing could help in the automated extraction and construction of multidimensional graphs from unstructured data such as text. For example, geo-location and sentiment analysis could be applied to enrich the attributes of the data entities and therefore equip businesses with the capability to analyze data from new perspectives. New graph entities could be discovered as well. For instance, using community detection, new labels could be added to the nodes, and using similarity, new edges could be added between the similar node. Multiple variation of the traditional ETL approach exist in the literature, and might be worth exploring for Graph ETL such as the Extract-Load-Transform, or the Capture-Transform-Flow.

- Graph Data Warehouse (3): The graph data warehouse is the reference central information repository for graph-based decision making. Data is extracted from different sources and integrated using a common graph model. The cleansed and integrated data is natively stored and managed as a multidimensional graph in the graph warehouse. Whereas that data would be transformed to tables in traditional data warehouses. Nevertheless, the conceptual layer remains the same (i.e., represented as dimensions and facts). The changes are related to the logical and physical levels. The graph warehouse provides a suitable backbone for natively analyzing graphs with BI tools such as graph OLAP and graph mining.
- Cube Design and Computation (4): The semantic relativism inherent in graphs allows to create several views from the same data and make them co-exist in a much simpler way than other data models. Afterwards, given a graph lattice, the graph cubes enable the computation and the aggregation of corresponding graph cuboids. Once the required graph cuboids are computed, the result is persisted in the corresponding data marts. To leverage graph properties, graph cubes embed graph-structured measures and dimensions. There is a need for cube computation and aggregation libraries capable of efficiently handling graphs. This line of research includes optimizations such as graph cuboids materialization, indexing, and graph icebergs.
- Discovery of multidimensional concepts and definition of potential multidimensional schemas (5): Multiple multidimensional schemas could be built from the same graph warehouse to satisfy the various analysis needs. Real-world graphs, such as social networks, are complex, dynamic and flexible. Interesting graph entities might be hidden in the large data sources. Therefore, there is need for novel graph-aware approaches that enables automatic detection and extraction of multidimensional concepts from large complex graphs. This could be done through the analysis of the topological aspects of graphs, and the projection of the properties of the multidimensional models on them. This will help end-users cope with the complexity and large volume of graphs, and expose potential interesting discovery to decision makers.
- Assistance with the analysis and synthesis of graphs (6): Given the complexity and large size of the initial graph, there is need for intelligent modules capable of performing automated preliminary analysis of the graph to guide the analyst during the exploration of the graph cubes. The goal is to enable

self-service BI and facilitate complex tasks such as the extraction of meaningful graph summaries, the discovery of interesting phenomena in the graph cuboids such as frequent graph patterns and outlier relationships.

- Mining and querying OLAP cubes (7–8): Complex and interactive OLAP analysis and mining of graph cubes is performed at this phase. In contrast to traditional OLAP, graph cubes enable the multidimensional analysis of graph metrics stored in the graph cuboids. For example, analysts could examine the centrality of leaders from multiple perspectives, or identify the communities and their connections at different levels of aggregations. To this end, there is a need to develop graph OLAP engines that support graph-structured cubes. In addition, Online Analytical Mining of graph data is a promising research direction to empower graph OLAP with mining capabilities. Graphs are dynamic and enabling OLAP on evolving networks by analyzing changing facts and dimensions will help in understanding the structural and informational evolution of networks. Many BI vendors have already integrated graphs into their BI solutions. However, the support for graphs is still limited and there is still a need to push further the integration of graph-derived insights into the decision-making process.

5 Conclusion

Graphs are interesting structures that provide a solid foundation for intuitively representing various domains and solving complex problems, while enabling better performance. Graph analytics leverage structural and content-based information to create added-value services, and extend current solutions with new topology-enabled capabilities. In this paper, we surveyed the state of the art on graph BI and analytics, and proposed an architecture of Graph BI and Analytics platform augmented with machine learning capabilities, which lays the foundations for promising future research directions. In all, graph analytics have a bright future, and this paper calls for more attention from academia and industry to build next-generation graph-powered BI and analytics frameworks.

References

1. Bean, R.: Variety, not volume, is driving big data initiatives (2016). <https://sloanreview.mit.edu/article/variety-not-volume-is-driving-big-data-initiatives/>. Accessed 25 Jan 2018
2. García-Solaco, M., Saltor, F., Castellanos, M.: In: Bukhres, O.A., Elmagarmid, A.K. (eds.) Object-Oriented Multidatabase Systems, pp. 129–202. Prentice Hall International (UK) Ltd, Hertfordshire, UK (1995)
3. Feinberg, D., Heudecker, N.: IT market clock for database management systems (2014). <https://www.gartner.com/doc/2852717/it-market-clock-database-management>. Accessed 02 Jan 2018
4. Akoglu, L., Tong, H., Koutra, D.: Graph based anomaly detection and description: a survey. *Data Mining Knowl. Discov.* **29**(3), 626–688 (2015)

5. Van Vlasselaer, V., Bravo, C., Caelen, O., Eliassi-Rad, T., Akoglu, L., Snoeck, M., Baesens, B.: Apatate: a novel approach for automated credit card transaction fraud detection using network-based extensions. *Decis. Support Syst.* **75**, 38–48 (2015)
6. Dasgupta, K., Singh, R., Viswanathan, B., Chakraborty, D., Mukherjea, S., Nanavati, A.A., Joshi, A.: Social ties and their relevance to churn in mobile telecom networks. In: *Proceedings of the 11th International Conference on Extending Database Technology, EDBT 2008. Advances in database technology*, New York, USA, pp. 668–677. ACM (2008)
7. Duan, L., Da Xu, L.: Business intelligence for enterprise systems: a survey. *IEEE Trans. Industr. Inform.* **8**(3), 679–687 (2012)
8. Lim, E.P., Chen, H., Chen, G.: Business intelligence and analytics: Research directions. *ACM Trans. Manag. Inf. Syst.* **3**(4), 17 (2013)
9. Cuzzocrea, A., Bellatreche, L., Song, I.Y.: Data warehousing and OLAP over big data: Current challenges and future research directions. In: *Proceedings of the Sixteenth International Workshop on Data Warehousing and OLAP*, pp. 67–70. ACM (2013)
10. Skhiri, S., Jouili, S.: Large graph mining: recent developments, challenges and potential solutions. In: *Aufaure, M.-A., Zimányi, E. (eds.) eBISS 2012. LNBIP*, vol. 138, pp. 103–124. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36318-4_5
11. Shi, C., Li, Y., Zhang, J., Sun, Y., Philip, S.Y.: A survey of heterogeneous information network analysis. *IEEE Trans. Knowl. Data Eng.* **29**(1), 17–37 (2017)
12. Chen, C., Yan, X., Zhu, F., Han, J., Yu, P.S.: Graph OLAP: a multi-dimensional framework for graph data analysis. *Knowl. Inf. Syst.* **21**(1), 41–63 (2009)
13. Hannachi, L., Benblidia, N., Boussaid, O., Bentayeb, F.: Community cube: a semantic framework for analysing social network data. *Int. J. Metadata Semant. Ontol.* **10**(3), 155–169 (2015)
14. Angles, R., Arenas, M., Barceló, P., Hogan, A., Reutter, J., Vrgoč, D.: Foundations of modern query languages for graph databases. *ACM Comput. Surv.* **50**(5), 68 (2017)
15. Hölsch, J., Schmidt, T., Grossniklaus, M.: On the performance of analytical and pattern matching graph queries in neo4j and a relational database. In: *Ioannidis, Y.E., Stoyanovich, J., Orsi, G. (eds.) Proceedings of the Workshops of the EDBT/ICDT 2017 Joint Conference (EDBT/ICDT 2017)*, Venice, Italy, March 21–24, 2017. Volume 1810 of CEUR Workshop Proceedings, CEUR-WS.org (2017)
16. Qu, Q., Zhu, F., Yan, X., Han, J., Yu, P.S., Li, H.: Efficient topological OLAP on information networks. In: *Yu, J.X., Kim, M.H., Unland, R. (eds.) DASFAA 2011, Part I. LNCS*, vol. 6587, pp. 389–403. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20149-3_29
17. Berlingerio, M., Coscia, M., Giannotti, F., Monreale, A., Pedreschi, D.: Multidimensional networks: foundations of structural analysis. *World Wide Web* **16**(5–6), 567–593 (2013)
18. Zhao, P., Li, X., Xin, D., Han, J.: Graph cube: On warehousing and OLAP multidimensional networks. In: *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pp. 853–864. ACM (2011)
19. Wang, Z., Fan, Q., Wang, H., Tan, K.L., Agrawal, D., El Abbadi, A.: Pagrol: Prallel Graph OLAP over large-scale attributed graphs. In: *2014 IEEE 30th International Conference on Data Engineering (ICDE)*, pp. 496–507. IEEE (2014)

20. Ghrab, A., Romero, O., Skhiri, S., Vaisman, A., Zimányi, E.: A framework for building OLAP cubes on graphs. In: Morzy, T., Valduriez, P., Bellatreche, L. (eds.) ADBIS 2015. LNCS, vol. 9282, pp. 92–105. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23135-8_7
21. Nebot, V., Berlanga, R.: Building data warehouses with semantic web data. *Decis. Support Syst.* **52**(4), 853–868 (2012)
22. Kämpgen, B., Harth, A.: Transforming statistical linked data for use in OLAP systems. In: Proceedings of the 7th International Conference on Semantic Systems, pp. 33–40. ACM (2011)
23. Beheshti, S.M.R., Benatallah, B., Motahari-Nezhad, H.R.: Scalable graph-based olap analytics over process execution data. *Distrib. Parallel Databases* **34**(3), 379–423 (2016)
24. Varga, J., Vaisman, A.A., Romero, O., Etcheverry, L., Pedersen, T.B., Thomsen, C.: Dimensional enrichment of statistical linked open data. *Web Semant. Sci. Serv. Agents World Wide Web* **40**, 22–51 (2016)
25. Nath, R.P.D., Hose, K., Pedersen, T.B., Romero, O.: SETL: a programmable semantic extract-transform-load framework for semantic data warehouses. *Inf. Syst.* **68**, 17–43 (2017)
26. Lee, K., Lee, K.: Escaping your comfort zone: a graph-based recommender system for finding novel recommendations among relevant items. *Expert Syst. with Appl.* **42**(10), 4851–4858 (2015)
27. Demesmaeker, F., Ghrab, A., Nijssen, S., Skhiri, S.: Discovering interesting patterns in large graph cubes. In: 2017 IEEE International Conference on Big Data (Big Data), pp. 3322–3331 (2017)
28. Bleco, D., Kotidis, Y.: Entropy-based selection of graph cuboids. In: Proceedings of the Fifth International Workshop on Graph Data-management Experiences & Systems, vol. 2. ACM (2017)
29. Lumsdaine, A., Gregor, D., Hendrickson, B., Berry, J.: Challenges in parallel graph processing. *Parallel Process. Lett.* **17**(01), 5–20 (2007)
30. Batarfi, O., El Shawi, R., Fayoumi, A.G., Nouri, R., Barnawi, A., Sakr, S., et al.: Large scale graph processing systems: survey and an experimental evaluation. *Cluster Comput.* **18**(3), 1189–1213 (2015)
31. Denis, B., Ghrab, A., Skhiri, S.: A distributed approach for graph-oriented multidimensional analysis. In: 2013 IEEE International Conference on Big Data, pp. 9–16, October 2013
32. Malewicz, G., Austern, M.H., Bik, A.J., Dehnert, J.C., Horn, I., Leiser, N., Czajkowski, G.: PREGEL: a system for large-scale graph processing. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, pp. 135–146. ACM (2010)
33. Low, Y., Bickson, D., Gonzalez, J., Guestrin, C., Kyrola, A., Hellerstein, J.M.: Distributed graphlab: a framework for machine learning and data mining in the cloud. *Proc. VLDB Endow.* **5**(8), 716–727 (2012)
34. Gonzalez, J.E., Low, Y., Gu, H., Bickson, D., Guestrin, C.: Powergraph: Distributed graph-parallel computation on natural graphs. In: OSDI, vol. 12, p. 2 (2012)
35. Gonzalez, J.E., Xin, R.S., Dave, A., Crankshaw, D., Franklin, M.J., Stoica, I.: Graphx: Graph processing in a distributed dataflow framework. *OSDI.* **14**, 599–613 (2014)
36. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets. In: Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing, HotCloud 2010, Berkeley, CA, USA, p. 10 (2010)

37. Junghanns, M., Petermann, A., Gómez, K., Rahm, E.: Gradoop: Scalable graph data management and analytics with hadoop. arXiv preprint [arXiv:1506.00548](https://arxiv.org/abs/1506.00548) (2015)
38. Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., Tzoumas, K.: Apache FLINK: Stream and batch processing in a single engine. In: Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, vol. 36(4) (2015)



Community Detection in Who-calls-Whom Social Networks

Ciprian-Octavian Truică^{1(✉)}, Olivera Novović², Sanja Brdar²,
and Apostolos N. Papadopoulos³

¹ Computer Science and Engineering Department, Faculty of Automatic Control
and Computers, University Politehnica of Bucharest, Bucharest, Romania

`ciprian.truica@cs.pub.ro`

² BioSense Institute, University of Novi Sad, Novi Sad, Serbia

`novovic@biosense.rs`, `brdars@uns.ac.rs`

³ Department of Informatics, Aristotle University of Thessaloniki,
Thessaloniki, Greece

`papadopo@csd.auth.gr`

Abstract. Mobile phone service providers collect large volumes of data all over the globe. Taking into account that significant information is recorded in these datasets, there is a great potential for knowledge discovery. Since the processing pipeline contains several important steps, like data preparation, transformation, knowledge discovery, a holistic approach is required in order to avoid costly ETL operations across different heterogeneous systems. In this work, we present a design and implementation of knowledge discovery from CDR mobile phone data, using the Apache Spark distributed engine. We focus on the community detection problem which is extremely challenging and it has many practical applications. We have used Apache Spark with the LOUVAIN community detection algorithm using a cluster of machines, to study the scalability and efficiency of the proposed methodology. The experimental evaluation is based on real-world mobile phone data.

Keywords: Data mining · Big data analytics · Community detection

1 Introduction

Mobile phone service providers collect large amount of data to monitor user interactions. Each time a user is using a mobile device (for sending an SMS or performing a call), a Call Detail Record (CDR) is created in the database of the service provider. Graphs have a central role in the analysis of mobile phone data collected by service providers. Due to their mathematical formalism and the variety of existing graph-based algorithmic techniques, they can be used efficiently and effectively in social networks, to solve specific problems.

C.-O. Truică and O. Novović contributed equally to this work.

© Springer Nature Switzerland AG 2018

C. Ordóñez and L. Bellatreche (Eds.): DaWaK 2018, LNCS 11031, pp. 19–33, 2018.

https://doi.org/10.1007/978-3-319-98539-8_2

Graph mining is a heavily active research direction with numerous applications [1]. One of the core research directions in the area is the discovery of meaningful communities in a large network [11]. In the majority of real-life applications, graphs are extremely sparse usually following power-law degree distribution. However, the original graph may contain groups of vertices, called *communities*, where vertices in the same community are more well-connected than vertices across communities.

The efficiency of community detection algorithms is heavily dependent on the size of the input graph, i.e., the number of vertices and/or the number of edges, and also on its structural complexity. In addition to the main processing task that must be performed, preprocessing is also a significant step that in many cases is computationally intensive. To handle both preprocessing and main processing efficiently, a potential solution is to use multiple resources and apply parallel or distributed computing techniques, aiming at reducing the overall processing time.

In this work, we focus on the analysis of real world CDR data, and more specifically on scalable community detection. CDRs are in general large in volume, and therefore scalable algorithmic techniques are required. In particular, we demonstrate the use of Apache Hadoop [26], Apache YARN [30], Apache Spark [15], and Apache Hive [28] in the mining process as a proof of concept.

In our previous work [20], we had used a conventional DBMS and Python to extract knowledge from raw telecom data. The experiments were very time consuming and we could analyze only a subset of the data in due time. The results that we obtained have motivated us to apply different processing techniques in order to speed up the experimental evaluation and to be able to analyze the complete dataset.

Our approach for analyzing the data is based on Apache Spark. The proposed implementation considers the complete pipeline, from preprocessing the raw data to knowledge discovery. All necessary tasks are executed within Spark and results are stored in Hive. Based on our performance evaluation results, we show that by applying graph sparsification through filtering, communities are discovered more efficiently, since runtime depends heavily on the number of edges in the graph. On the other hand, by comparing the communities generated with and without filtering we observe that communities remain relatively stable in comparison to the ground truth (unfiltered graph). The discovered communities reflect dynamic social interactions that are along with other components of the city, transport and land use, essential for the understanding of such a complex system [10, 13].

The rest of the paper is organized as follows. In the next section we describe briefly related work in the area. The proposed methodology is described in detail in Sect. 3. Section 4 presents the implementation of our pipeline. Section 5 offers performance evaluation results using real-world networks. Finally, Sect. 6 concludes the work and presents briefly some interesting future research directions.

2 Related Work

During the last few years, there is a tremendous growth of new applications that are based on the analysis of mobile phone data [6]. Among them there are many applications with significant societal impact such as, urban sensing and planning [5, 10], traffic engineering [2, 14], predicting energy consumption [8], disaster management [18, 22, 33], epidemiology [9, 17, 32], deriving socio-economical indicators [21, 27].

To enable development and run of applications and services on such data, current efforts are directed toward providing access to these large scale human behavioral data in a privacy-preserving manner. Recent initiative of Open Algorithms (OPAL) has suggested approach of moving the algorithm to the data [16]. In this model, raw data are never exposed to outside parties, only vetted algorithms run on telecom companies' servers. This poses huge challenge on efficient processing of data, especially when array of parties is interested in extracting information and getting insights from data.

A significant graph mining task with important applications is the discovery of meaningful communities [11]. In many cases, community detection is solved by using graph clustering algorithms. However, a major limitation of graph clustering algorithms is that they are computationally intensive, and therefore their performance deteriorate rapidly, as we increase the size of the data.

An algorithm that has been used for community detection in large networks is the LOUVAIN algorithm, proposed in [7]. This algorithm has many practical applications and it scales well with the size of the data. Moreover, it has been used in several studies related to static or evolving community detection [3].

In this work, we combine the efficiency of the LOUVAIN algorithm with the power of the Apache Spark distributed engine, to demonstrate that we can support the full pipeline of community detection in a efficient manner.

3 Proposed Methodology

In this section, we describe our approach in detail, explaining the algorithms for community detection and filtering as well as the evaluation methods used.

3.1 Graph Mining and Community Detection

Among the different existing graph mining problems, we center our focus on community detection. The graph, in our case, corresponds to user interactions aggregated to Radio Base Station (RBS) level. Therefore, communities correspond to groups of RBSs with strong pair-wise activity within each group. To enable efficient community detection in potentially massive amounts of data, we need to attack the following problems: (i) the algorithmic techniques applied must scale well with respect to the size of the data, which means that the algorithmic complexity should stay below $\mathcal{O}(n^2)$ (where n is the number of graph nodes), and (ii) since we do not know the number of communities in advance,

the algorithms used must be flexible enough to be able to infer the number of communities during the course of the algorithm.

To meet the aforementioned requirements, we have chosen to apply a modularity-based algorithm proposed in [7]. The concept of modularity [19] presented by Eq. (1), is used to estimate the quality of the partitions, where A_{ij} is the weight of the edge connecting the i -th and the j -th node of the graph, $\sum_j A_{ij}$ is the sum of the weights of the edges attached to the i -th node, c_i is the community where the i -th node is assigned to, $m = (1/2) \sum_{i,j} A_{ij}$, and $\delta(x, y)$ is zero if nodes x and y are assigned to the same community and 1 otherwise.

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{i,j} - \frac{\sum_j A_{ij} \cdot \sum_i A_{ji}}{2m} \right] \delta(c_i, c_j) \quad (1)$$

Unfortunately, computing communities based on the maximization of the modularity, is an \mathcal{NP} -hard problem. To provide an efficient solution, the algorithm proposed in [7] uses an iterative process that involves shrinking the graph, every time modularity converges. In each phase, each node is assigned to a neighboring community that maximizes the modularity of the graph. As long as nodes are moving around communities and modularity grows, we keep on executing this process. When there are no more changes, a shrinking process is executed. Upon shrinking the graph, each community produced during the previous phase is assigned to the same super node of the new graph. Then, the same technique is applied to the new graph. The algorithm terminates when the modularity detected in the new graph is less than the modularity detected in the previous graph. The set of communities that maximize the modularity is returned as an answer. The outline of the technique is depicted in Algorithm 1.

The network we are studying is undirected and weighted, but elimination of some edges by simple thresholding the weight values is not going to reveal the core backbone of the network. Moreover, we needed a method for graph filtering that will consider local properties of the nodes, such as weight over all edges linked to specific node. To meet the aforementioned requirements we have chosen to apply the disparity filter [25]. The disparity filter uses the null model to define significant links, where the null hypothesis states that weighted connections of the observed node are produced by a random assignment from a uniform distribution. The disparity filter proceeds by identifying strong and weak links for each node. The discrimination is done by calculating for each edge the probability α_{ij} that its normalized weight p_{ij} is compatible with the null hypothesis. All the links with $\alpha_{ij} < \alpha$ reject the null hypothesis. The statistically relevant edges will be those whose weights satisfy the Eq. (2).

$$\alpha_{ij} = 1 - (n - 1) \int_0^{p_{ij}} (1 - x)^{n-2} dx < \alpha \quad (2)$$

We note that smaller values of α_{ij} denote more significant edges. Therefore, filtering is applied by keeping all edges where $\alpha_{ij} \leq \alpha$ and thus removing all edges where $\alpha_{ij} > \alpha$. By changing the alpha threshold value we can filter out the links

Algorithm 1. LOUVAIN ($G(V, E)$)

```

Input: the graph  $G$ 
Result: the communities of  $G$ 
1  $n \leftarrow |V|$  /* number of graph nodes */
2  $done \leftarrow false$ 
3 while not  $done$  do
4   assign each  $u \in V$  to a different community
5   while there is a change do
6     for every node  $u \in V$  do
7        $C \leftarrow$  a community that maximizes modularity /*  $C$  is a
          neighboring community or  $u$ 's community */
8     if  $newModularity > oldModularity$  then
9        $G \leftarrow$  shrink graph based on communities /* each community
          becomes a super node in the new graph */
10    else
11      return communities

```

with small significance focusing on more relevant edges. The α_{ij} represent the statistical probability, so its value is in range $[0, 1]$. The threshold alpha value is set in regards to the significance level with which we want to apply the filtering. In our experiments, we applied three different filtering levels with probability 95%, 99% and 99.9%, where corresponding α values are 0.05, 0.01 and 0.001 respectively.

3.2 Clustering Evaluation Methods

To evaluate the community detection, we will use classical cluster evaluation methods, i.e. Purity, Entropy, Rand Index and Adjusted Rand Index.

The *Purity* (Definition 1) of a cluster measures the extent to which each cluster contains elements from primarily one class [34].

Definition 1 (Purity). *Given a set S of size n and a set of cluster C of size k , then, for a cluster $c_i \in C$ of size k_i , the purity is $p(c_i) = \frac{\max_i(n_j^i)}{k_i}$, where n_j^i is the number of elements of the j -th class assigned to the i -th cluster. The overall purity is defined as $P(C) = \sum_{i=1}^k \frac{k_i}{n} \cdot p(c_i)$.*

Entropy (Definition 2) is an evaluation method that assumes that all elements of a set have the same probability of being picked and, by choosing an element at random, the probability of this element to be in a cluster can be computed [31].

Definition 2 (Entropy). *Given a set S of size n and a set of clusters C of size k , then, by assuming that all elements in S have the same probability of being picked, the probability of an element $s \in S$ chosen at random to belong to*

cluster $c_i \in C$ of size k_i is $p(c_i) = \frac{k_i}{n}$. Then, the overall entropy associated with C is $H(C) = -\sum_{i=1}^k p(c_i) \cdot \log_2(p(c_i))$.

The *Rand Index (RI)* (Definition 3) is a measure used to determine the similarity between two data clusterings [23].

Definition 3 (Rand Index). Given a set $S = \{s_1, s_2, \dots, s_n\}$ of n elements and two groupings $X = \{X_1, X_2, \dots, X_r\}$ and $Y = \{Y_1, Y_2, \dots, Y_t\}$ then the Rand Index is $RI = \frac{a+b}{\binom{n}{2}}$, where $a = |S'|$ with $S' = \{(s_i, s_j) | s_i, s_j \in X_k, s_i, s_j \in Y_l\}$ and $b = |S''|$ with $S'' = \{(s_i, s_j) | s_i \in X_{k_1}, s_j \in X_{k_2}, s_i \in Y_{l_1}, s_j \in Y_{l_2}\}$ for some $1 \leq i, j \leq n, i \neq j, 1 \leq k, k_1, k_2 \leq r, k_1 \neq k_2, 1 \leq l, l_1, l_2 \leq t, l_1 \neq l_2$.

Adjusted Rand Index (ARI) (Definition 4) is a cluster evaluation method that calculates the fraction of correctly classified (respectively misclassified) elements to all elements by assuming a generalized hypergeometric distribution as null hypothesis [31]. ARI is the normalized difference of the Rand Index and its expected value under the null hypothesis [29]. ARI uses the contingency table.

Definition 4 (Adjusted Rand Index). Given a set S of n elements and two groupings $X = \{X_1, X_2, \dots, X_r\}$ and $Y = \{Y_1, Y_2, \dots, Y_s\}$, the overlap between X and Y , can be summarized in a contingency table $[n_{ij}]$, where $n_{ij} = |X_i \cap Y_j|$, $a_i = \sum_{j=1}^s n_{ij}$ and $b_j = \sum_{i=1}^r n_{ij}$. Using the contingency table, the Adjusted Rand Index is defined in Eq. (3).

$$ARI = \frac{\sum_{i,j} \binom{n_{ij}}{2} - \frac{\sum_i \binom{a_i}{2} \cdot \sum_j \binom{b_j}{2}}{\binom{n}{2}}}{\frac{1}{2}(\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}) - \frac{\sum_i \binom{a_i}{2} \cdot \sum_j \binom{b_j}{2}}{\binom{n}{2}}} \quad (3)$$

4 Implementation Methodology




4.1 Dataset

Telecommunication interaction between mobile phone users is managed by Radio Base Stations (RBS) that are assigned by the operator. Every RBS has a *unique id*, a *location* and a *coverage map* that provides approximate user's geographical location. CDRs contain the time of the interaction and the RBS which handled it. In available data collection CDRs are spatially aggregated on the grid containing 10,000 cells and temporally aggregated on time slots of ten minutes. Evidently, the final network is composed of 10,000 nodes. However, if we change the granularity and the level of the detail as well as the geographical area that we are interested in, the number of nodes can grow easily to millions.

Community detection is done using the LOUVAIN algorithm. Filtering is used to minimize the graph and keep only the important nodes without losing the communities. In our case, the number of nodes is 10,000. The value of the parameter α_j (Eq. (2)) defines the level of filtering. By changing the filtering level we

obtain graphs with fewer edges. We are using these graphs to demonstrate the performance of community detection as we grow the number of edges in the input graph.

The dataset (C) provides real world information regarding the *directional interaction strength* (DIS) based on call exchanged between different areas of the city of Milan [4] and it's publicly available online¹. The DIS between two areas ($SID1$ and $SID2$) is proportional to the number of calls issued from area $SID1$ to area $SID2$. The temporal values, given as a timestamp, are aggregated in time slots which represent the beginning of the interval. The dataset represents a directed graph. For our experiments, two transformations are applied on the original corpus: (i) the directed graph is transformed into an undirected graph, i.e. the edges (SID_i, SID_j) and (SID_j, SID_i) are going to be represented as a single edge (SID_i, SID_j) with the edge $Cost_{ij} = DIS_{ij} + DIS_{ji}$ for the same timestamp, and (ii) the timestamp is aggregated to a calendar date ($Date$), i.e. for an edge (SID_i, SID_j) for each $Date$ the cost is $Cost = \sum DIS_{ij}$. Using this information we compute for each edge the parameter α_{ij} using Eq. (2). Moreover, an *Edge Cost Factor* (ECF) is used to normalize the values for the edges' $Cost \in [9 \cdot 10^{-13}, 466]$ when applying LOUVAIN. This information is stored in a Hive database installed on top of Hadoop's HDFS and MapReduce. Figure 1 presents the database diagram where the information about edges is stored in the *Edges* (E) table, while the information about communities is stored in the *Louvain* (L) table, where *Level* is the algorithm's iterations.

Edges		
	Date	date
	SID1	integer
	SID2	integer
	Cost	float
	Alpha	float





Louvain		
	Date	date
	SID	integer
	pAlpha	float
	ECF	integer
	Level	integer
	Comunity	integer

Fig. 1. The Hive database schema.

4.2 System Architecture

Apache Spark is a unified distributed engine with a rich and powerful APIs for different programming languages [15], i.e. Scala, Python, Java and R. One of its main characteristics is that (in contrast to Hadoop MapReduce) it exploits main memory as much as possible, being able to persist data across rounds to avoid unnecessary I/O operations. Spark jobs are executed based on a master-slave model in cooperation with the cluster manager YARN.

¹ Dataset <http://theodi.fbk.eu/openbigdata/>.

The proposed architecture is based on the Apache Spark engine using the Scala programming language together with the Spark Dataframes, HiveContext and GraphX libraries. The information is stored in an Apache Hive database which is installed on top of HDFS. The cluster resource manager used is YARN. Our methodology utilizes the following pipeline (Fig. 2):

1. CDRs are aggregated in such a way that each graph node corresponds to a spatial area. This task has been performed by the mobile operator before releasing the data.
2. The original directed graph is aggregated to obtain an undirected one, as the orientation of edges is ignored in our case.
3. Filtering is applied in order to sparsify the network.
4. Community detection is applied using the LOUVAIN algorithm.
5. Visualization is applied.

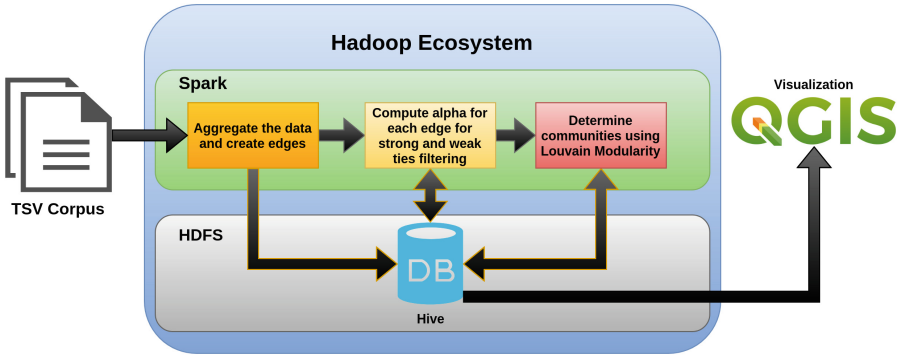


Fig. 2. Architecture

4.3 Queries

The Table of Edges. To create the edges (E), the DIS for attributes $DATE$, $SID1$ and $SID2$, which represent a compound primary key that uniquely identify each record, is aggregated. The aggregation constructs an undirected graph using the union between the records where $SID1 \leq SID2$ and $SID1 > SID2$. The resulting graph is stored in the Hive database. The following query, expressed in relational algebra, is used to populate the $Edges$ table:

$$E = \rho_{\frac{Cost}{F_0}}(\pi_{Date, SID1, SID2, F_0}(\gamma_{L_0}(\pi_{Date, SID1, SID2, DIS}(\sigma_{SID1 \leq SID2}(C)) \uplus \pi_{Date, SID2, SID1, DIS}(\sigma_{SID1 > SID2}(C)))))$$

where C is the corpus and γ_{L_0} is the aggregation operator with $L_0 = (F_0, G_0)$, $F_0 = sum(DIS) = Cost$, the sum is the aggregation function that sums the

DIS for all the pairs $(Date, SID1, SID2)$ and $G_0 = (Date, SID1, SID2)$ is the list of attributes in the GROUP BY clause.

Strong and Weak Ties. To determine the strong and weak ties for filtering, α_{ij} from Eq. (2) is computed for each unique key $(Date, SID1, SID2)$. Therefore, the integral must be solved in order to use directly the information stored in the database and to improve the time performance: $\alpha = 1 - (k-1) \cdot \int_0^{p_{ij}} (1-x)^{k-2} dx = 1 - (1-p_{ij})^{k-1}$, where $p_{ij} = \frac{Cost_{ij}}{\sum_{j,i \neq j} Cost_{ij}} = \frac{Cost}{\sum_c} = Alpha$ and $k = N$ the number of nodes.

To compute the sum of costs for each $Date$ and $SID1$ (\sum_c), the following query, given in relational algebra, is used:

$$R_1 = \rho_{\frac{\sum_c}{F_1}}(\pi_{Date, SID1, F_1}(\gamma_{L_1}(\sigma_{SID1 \neq SID2}(E))))$$

where γ_{L_1} is the aggregation operator with $L_1 = (F_1, G_1)$, $F_1 = sum(Cost) = \sum_c$, the sum is the aggregation function that sums the $Cost$ for all the pairs $(Date, SID1)$, and $G_1 = (Date, SID1)$ is the list of attributes in the GROUP BY clause.

The following query is used to compute the number of distinct nodes for each $Date$:

$$R_2 = \rho_{\frac{N}{F_2}}(\pi_{Date, SID1, F_2}(\gamma_{L_2}(E)))$$

where γ_{L_2} is the aggregation operator with $L_2 = (F_2, G_2)$, $F_2 = count(DISTINCT SID1) = N$, the $count$ is the aggregation function that counts the distinct number of nodes for a $Date$ and $G_2 = (Date)$ is the list of attributes in the GROUP BY clause.

To compute α_{ij} , R_1 must be joined with R_2 . The result is stored in the database either as a separate table or by updating the $Edges$'s table column $Alpha$. The query expressed in relational algebra is:

$$A = \pi_{Date, SID1, SID2, Cost, Alpha}(\sigma_{SID1 \neq SID2}(E) \bowtie_{\theta} R_1 \bowtie_{E.Date=R_2.Date} R_2)$$

The θ condition in the first join operator is defined as: $E.Date = R_1.Date \wedge E.SID1 = R_1.SID1$.

Community Detection. To apply the LOUVAIN algorithm and detect communities for a given date ($pDate$) and alpha threshold ($pAlpha$), the following query is used to extract the information from the database:

$$L_M = \pi_{SID1, SID2, Cost}(\sigma_{Date=pDate \wedge Alpha \leq pAlpha}(A))$$

The LOUVAIN algorithm is implemented in the Spark engine using the Scala programming language and the GraphX, Dataframes and HiveContext library. Our implementation extends an existing implementation² so that it works with our architecture. In order to apply LOUVAIN, a graph should be given in the input, i.e., using the query L_M . The result of the algorithm is stored in the Hive database in table *Louvain*.

² Louvain <https://github.com/Sotera/spark-distributed-louvain-modularity>.

5 Performance Evaluation

The goal of the evaluation is to demonstrate the performance of community detection with and without filtering, using Apache Spark. Two diverse systems have been used: (i) a multi core server machine running CentOS with 16 Intel Xeon E5-2623 CPUs with 4 cores at 2.60 GHz, 126 GB RAM and 1 TB HDD, and (ii) a cluster with 6 nodes running Ubuntu 16.04 \times 64, each with 1 Intel Core i7-4790S CPU with 8 cores at 3.20 GHz, 16 GB RAM and 500 GB HDD. The Hadoop ecosystem is running on Ambari and has the following configuration for the 6 nodes: 1 node acts as HDFS NameNode and SecondaryName Node, YARN ResourceManager, and Hive Metastore and 5 nodes, each acting as HDFS DataNodes, YARN NodeManagers, Spark Client, and Hive Client.

For the experiments we fix the number of Spark executors to 16 with one vnode and 3 GB memory each. The same settings have been used in both the Single Machine and Cluster Mode. The complete pipeline has been implemented in Scala, whereas the code is publicly available on GitHub³.

5.1 Runtime Evaluation

The first experiment measures the performance of edge generation from raw data. The second experiment measures the performance of computing α values (see Eq. (2)). The third experiment is related to the performance of LOUVAIN algorithm over a filtered graph where $\alpha = 0.05$ and $ECF = 1,000,000$.

For each experiment we measure the average runtime over a series of 10 runs. For the first task (creating edges), the cluster environment showed 42% better runtime than the single mode case, whereas the standard deviation is relatively small in both cases. For the task related to computing α values, the cluster showed again better performance, since the computation is 63% faster. In this case, the standard deviation is higher for both environments in comparison to the previous task. Finally, the performance of community detection using LOUVAIN differ for each graph, but in most cases the cluster showed the best performance. In most of the cases the mean values of runtime for LOUVAIN are higher in the case of single machine, whereas the standard deviation is higher in the cluster environment. We hypothesize that these results are a direct consequence of how YARN’s ResourceManager schedules the ApplicationManager and NodeManager and Spark’s directed acyclic graph (DAG) execution engine optimizes the graph construction for the GraphX library. Figure 3 shows some representative comparative results.

5.2 Community Detection and Evaluation

For community visualization, the QGIS software has been used. We have chosen to present the set of communities generated by using the 8th of November, because the network for this particular day contains the highest number of edges.

³ GitHub repository https://github.com/cipriantruica/MBD_CommunityDetection.

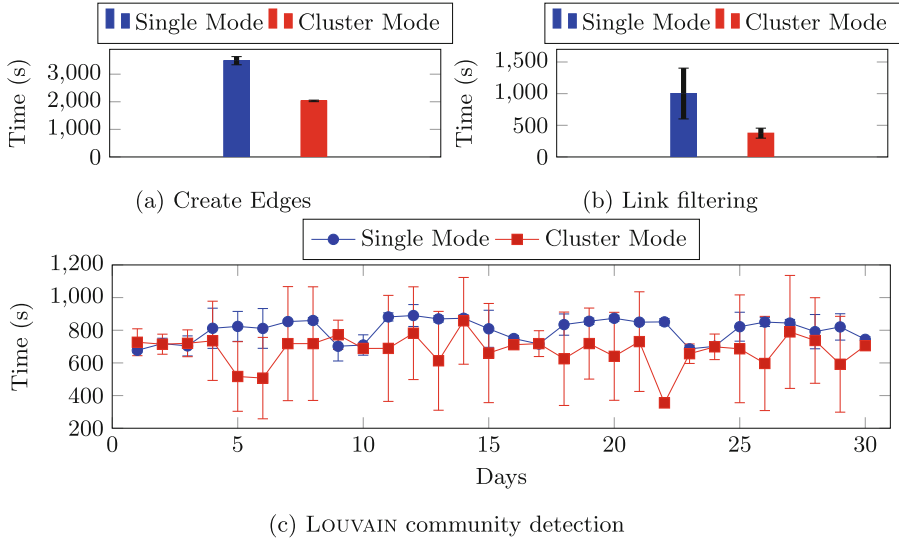


Fig. 3. Runtime results.

Experiments are performed on the cluster for three different threshold values, i.e., $\alpha = \{0.001, 0.01, 0.05\}$, and $ECF = 10^{12}$. We used the results of community detection performed over unfiltered graph as the ground truth result, and compare it with the results when the filtering was applied. After filtering is performed, the LOUVAIN community detection algorithm is executed on Spark. The first level of filtering eliminates more than 50% of edges, while the runtime for LOUVAIN clustering algorithm improves with a factor of 2.46. When $\alpha = 0.01$ almost 70% of edges are eliminated, and the algorithm runtime improves with a factor of 3.7. Filtering with $\alpha = 0.001$ eliminates almost 80% of the edges, the algorithm’s runtime improves by a factor of 6.88.

Table 1. Number of nodes and edges after applying different filtering levels, number of communities and runtime community detection.

α	# nodes	# edges	# communities	Time (sec.)
1	10,000	29,099,392	175	$2,229.73 \pm 340.14$
0.05	10,000	12,942,551	174	906.43 ± 279.79
0.01	10,000	9,003,404	176	603.20 ± 174.28
0.001	10,000	6,043,769	186	324.21 ± 127.73

The number of communities changes when filtering is applied. The results for 10 tests are presented in Table 1. The LOUVAIN community detection algorithm converges to the same result for the same input graph. The number of communities is higher for the graphs where the filtering is stricter. That is expected,

because the higher level of filtering gets the graph containing the strongest links. Moreover, as the number of nodes stays constant, the removal of edges tends to create more communities. In Fig. 4 we observe the centrality pattern of the clustering for each graph. The structure of communities is denser in the city center area, while in the peripheral parts communities are more spatially spread. That is due to overall higher traffic of people in city center, which reflects to telecom network. We observe also that the number of communities produced from graphs where the filtering with $\alpha = 0.01$ and $\alpha = 0.05$ is applied does not differ much from the number of communities produced from the unfiltered graph. On the other hand, the runtime for filtered and unfiltered graphs differs significantly. Even with the less strict filtering applied, we get a major improvement in processing time, which justifies the use of filtering.

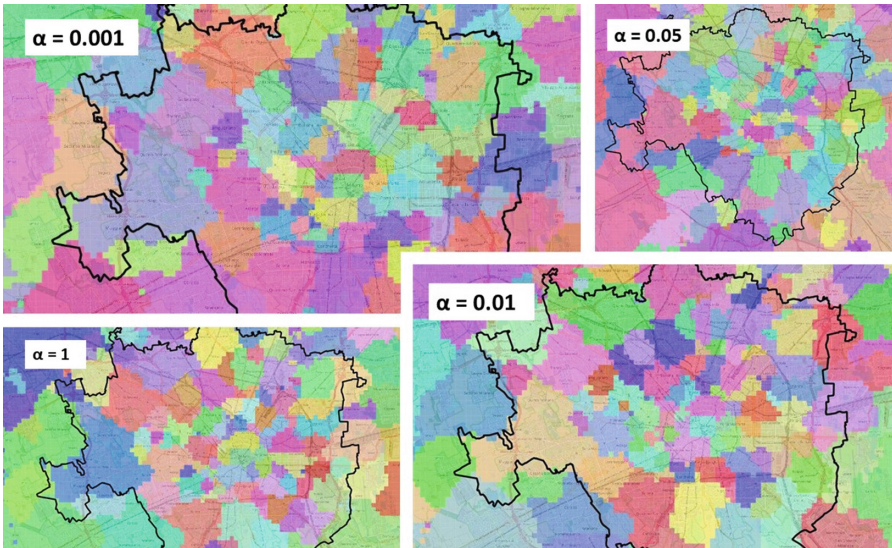


Fig. 4. Communities formed for different filtering thresholds.

Community evaluation is performed using *Purity*, *Entropy*, *Rand Index* and *Adjusted Rand Index* measures. The results are given in Table 2 and they are

Table 2. Clustering evaluation for different threshold values.

Evaluation measure	$\alpha = 0.001$	$\alpha = 0.01$	$\alpha = 0.05$
Purity	0.055	0.052	0.048
Entropy	0.902	0.909	0.912
Rand Index	0.981	0.985	0.987
Adjusted Rand Index	0.662	0.745	0.781

obtained by using LOUVAIN without filtering as the ground truth. *Purity* measures how many elements from the communities determined by LOUVAIN, when filtering is applied belong, to the ground truth communities. This measure is low because the number of communities differ between the ground truth and each different α threshold. The *Entropy* measures the probability of a node chosen at random to belong to a community. The high results of this measure describes how much we can, on the average, reduce the uncertainty about the cluster of a random element when knowing its cluster, the ground truth, in another clustering of the same set of elements. The *Rand Index* provides information on how a new clustering is compared to a correct one, the ground truth, and it is highly dependent on the number of clusters. This measure is high for all the tests because the *Rand Index* converges to 1 as the number of clusters increases which is undesirable for a similarity measure [12]. To address this problem we also computed the *Adjusted Rand Index* which shows a clearer classification for the filtering technique. For the threshold $\alpha = 0.05$, as well as for $\alpha = 0.01$, the number of communities remain stable and closer to the ones detected in the ground truth, although the number of edges decreases significantly (see Table 1).

6 Conclusions

Mobile phone records offer many potentials for knowledge discovery with significant impact. In particular, community detection is a task related to networks and aims at the discovery of groups of nodes that are densely connected. In general, community detection is solved by executing graph clustering algorithms, which is a computationally intensive task, and therefore scalable algorithms should be applied to guarantee efficiency for large networks.

This work focuses on community detection in a distributed environment, based on real-world mobile phone data. The first results has shown that parallelism is an important tool to attack scalability issues, since we can analyze larger graphs by using a cluster of machines. Moreover, we have shown that by applying sparsification through filtering, we may boost performance even further without penalizing the quality of the community detection result.

There are several interesting directions for future work, such as: *(i)* the modification of the modularity definition to reflect the association between CDR records and spatial information, *(ii)* the implementation and comparison of different community detection algorithms (e.g., Infomap [24]) and filtering techniques, *(iii)* the impact of filtering on other community detection algorithms, and *(iv)* the design of distributed community detection techniques for evolving networks, combining the mobile data information with the spatial location and tracking changes in communities as the network evolves. In addition, we are planning to test the proposed methodology for massive real-world networks, to be able to study scalability more thoroughly.

Acknowledgement. This article is based upon work from COST Action IC1406 High-Performance Modelling and Simulation for Big Data Applications (cHiPSet), supported by COST (European Cooperation in Science and Technology).

References

1. Aggarwal, C.C., Wang, H.: *Managing and Mining Graph Data*. Springer, London (2010)
2. Alexander, L., Jiang, S., Murga, M., González, M.C.: Origin-destination trips by purpose and time of day inferred from mobile phone data. *Transp. Res. Part C Emerg. Technol.* **58**, 240–250 (2015)
3. Aynaud, T., Guillaume, J.: Static community detection algorithms for evolving networks. In: *International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, pp. 513–519 (2010)
4. Barlacchi, G., De Nadai, M., Larcher, R., Casella, A., Chitic, C., Torrisi, G., Antonelli, F., Vespignani, A., Pentland, A., Lepri, B.: A multi-source dataset of urban life in the city of Milan and the Province of Trentino. *Sci. Data* **2** (2015). Article ID 150055
5. Becker, R.A., Caceres, R., Hanson, K., Loh, J.M., Urbanek, S., Varshavsky, A., Volinsky, C.: A tale of one city: using cellular network data for urban planning. *IEEE Pervasive Comput.* **10**(4), 18–26 (2011)
6. Blondel, V.D., Decuyper, A., Krings, G.: A survey of results on mobile phone datasets analysis. *EPJ Data Sci.* **4**(1), 10 (2015)
7. Blondel, V.D., Guillaume, J.-L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. *J. Stat. Mech. Theor. Exp.* **2008**(10) (2008). Article ID P10008
8. Bogomolov, A., Lepri, B., Larcher, R., Antonelli, F., Pianesi, F., Pentland, A.: Energy consumption prediction using people dynamics derived from cellular network data. *EPJ Data Sci.* **5**(1), 13 (2016)
9. Brdar, S., Gavrić, K., Čulibrk, D., Crnojević, V.: Unveiling spatial epidemiology of HIV with mobile phone data. *Sci. Rep.* **6**, article id 19342 (2016)
10. Calabrese, F., Ferrari, L., Blondel, V.D.: Urban sensing using mobile phone network data: a survey of research. *ACM Comput. Surv.* **47**(2), 25:1–25:20 (2014)
11. Fortunato, S.: Community detection in graphs. *Phys. Rep.* **483**(3), 75–174 (2010)
12. Fowlkes, E.B., Mallows, C.L.: A method for comparing two hierarchical clusterings. *J. Am. Stat. Assoc.* **78**(383), 553–569 (1983)
13. Gao, S., Liu, Y., Wang, Y., Ma, X.: Discovering spatial interaction communities from mobile phone data. *Trans. GIS* **17**(3), 463–481 (2013)
14. Järv, O., Ahas, R., Saluveer, E., Derudder, B., Witlox, F.: Mobile phones in a traffic flow: a geographical perspective to evening rush hour traffic analysis using call detail records. *PLoS ONE* **7**(11), 1–12 (2012)
15. Karau, H., Konwinski, A., Wendell, P., Zaharia, M.: *Learning Spark: Lightning-Fast Big Data Analytics*, 1st edn. O’Reilly Media Inc., Sebastopol (2015)
16. Lepri, B., Oliver, N., Letouzé, E., Pentland, A., Vinck, P.: Fair, transparent, and accountable algorithmic decision-making processes. *Philos. Technol.* 1–17 (2017)
17. Lima, A., De Domenico, M., Pejovic, V., Musolesi, M.: Disease containment strategies based on mobility and information dissemination. *Sci. Rep.* **5**, article id 10650 (2015)
18. Lu, X., et al.: Detecting climate adaptation with mobile network data in Bangladesh: anomalies in communication, mobility and consumption patterns during cyclone Mahasen. *Clim. Change* **138**(3–4), 505–519 (2016)
19. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. *Phys. Rev. E* **69** (2004). Article ID 026113

20. Novović, O., Brdar, S., Crnojević, V.: Evolving connectivity graphs in mobile phone data. In: NetMob, The Main Conference on the Scientific Analysis of Mobile Phone Datasets, pp. 73–75. Vodafone (2015)
21. Pappalardo, L., Pedreschi, D., Smoreda, Z., Giannotti, F.: Using big data to study the link between human mobility and socio-economic development. In: IEEE International Conference on Big Data (Big Data), pp. 871–878 (2015)
22. Pastor-Escuredo, D., Morales-Guzmán, A., Torres-Fernández, Y., Bauer, J.-M., Wadhwa, A., Castro-Correa, C., Romanoff, L., Lee, J.G., Rutherford, A., Frias-Martinez, V., Oliver, N., Frias-Martinez, E., Luengo-Oroz, M.: Flooding through the lens of mobile phone activity. In: Global Humanitarian Technology Conference (GHTC), pp. 279–286. IEEE, October 2014
23. Rand, W.M.: Objective criteria for the evaluation of clustering methods. *J. Am. Stat. Assoc.* **66**(336), 846–850 (1971)
24. Rosvall, M., Bergstrom, C.T.: Maps of random walks on complex networks reveal community structure. *Proc. Natl. Acad. Sci.* **105**(4), 1118–1123 (2008)
25. Serrano, M.Á., Boguná, M., Vespignani, A.: Extracting the multiscale backbone of complex weighted networks. *Proc. Natl. Acad. Sci.* **106**(16), 6483–6488 (2009)
26. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The Hadoop distributed file system. In: Symposium on Mass Storage Systems and Technologies, pp. 1–10 (2010)
27. Steele, J.E., et al.: Mapping poverty using mobile phone and satellite data. *J. R. Soc. Interface* **14**(127), article id 20160690 (2017)
28. Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Anthony, S., Liu, H., Wyckoff, P., Murthy, R.: Hive: a warehousing solution over a map-reduce framework. *Proc. VLDB Endow.* **2**(2), 1626–1629 (2009)
29. Truică, C.-O., Rădulescu, F., Boicea, A.: Comparing different term weighting schemas for topic modeling. In: International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2016), pp. 307–310 (2016)
30. Vavilapalli, V.K., et al.: Apache Hadoop YARN: yet another resource negotiator. In: Annual Symposium on Cloud Computing, pp. 5:1–5:16 (2013)
31. Wagner, S., Wagner, D.: Comparing clusterings: an overview. Technical report (2007)
32. Wesolowski, A., et al.: Impact of human mobility on the emergence of dengue epidemics in Pakistan. *Proc. Natl. Acad. Sci.* **112**(38), 11887–11892 (2015)
33. Wilson, R., et al.: Rapid and near real-time assessments of population displacement using mobile phone data following disasters: the 2015 Nepal earthquake. *PLoS Curr.* **8** (2016)
34. Zhao, Y., Karypis, G.: Criterion functions for document clustering: experiments and analysis. Technical report (2002)



FedS: Towards Traversing Federated RDF Graphs

Qaiser Mehmood^(✉), Alok Kumar Jha, Dietrich Rebholz-Schuhmann,
and Ratnesh Sahay

Insight Centre for Data Analytics, National University of Ireland, Galway, Ireland
{qaiser.mehmood, alok.kumar, rebholz, ratnesh.sahay}@insight-centre.org

Abstract. Traversing paths within a graph is a well-studied problem and highly intractable especially with large-scale graphs. In case of multiple graphs, the standard practice is to merge distinct graphs in a centralised way to evaluate the existence of paths between given entities (or nodes). In the biomedical domain counting and retrieving the number of paths (or edges) that connect two biological entities is a highly desirable feature expected from graph databases. Therefore, non-standard solutions exist that count and retrieve paths from a single graph database. From the standard perspective, SPARQL 1.1 provides the navigational feature called Property Paths (PP) which is limited only to a single RDF graph where path existence can be evaluated between pair of nodes. In this paper, we propose a federated approach – called FedS – that retrieves paths from multiple RDF triple stores. Our key idea is to partially delegate computational load to a set of federated RDF triple stores in a peer-to-peer manner thus reducing the computational burden on a centralised query processing server. In our preliminary investigation, we evaluate FedS against the state-of-the-art approaches that provide the path counting feature over single RDF graph. We compare FedS against these approaches in terms of performance (overall path retrieval time) and result completeness, i.e., number of paths retrieved.

1 Introduction

It is very common in the biomedical domain that two biological entities (gene, protein, pathway, drug, etc.) are associated via several properties and therefore, discovery of such connecting properties (or paths) between two given biological entities is often a fundamental requirement behind novel biological innovations [5, 12]. The recent expansion of “Web of Data”, particularly the life sciences portion of the Link Open Data (LOD) Cloud¹, has thrown entirely new dimensions of challenges for the graph-based knowledge and data integration communities. The life-sciences community has been very active within the Linked Open Data (LOD) movement: 41 datasets on the LOD cloud are classified as specialising in the “Life Sciences” domain and 70 SPARQL endpoints have been

¹ <http://lod-cloud.net/>.

made available by these publishers, most prominently by the Bio2RDF² [1,2] and Linked Life Data³ initiatives. Other general-knowledge datasets, such as DBpedia and corresponding triple stores, also contain rich information about the life sciences. Although, the underlying SPARQL endpoints behind the LOD Cloud has created a mashup of linked data connected through various linking properties (e.g., *xRef*, *owl:sameAs*, *x-relation*) [9]. Unfortunately, there is no mechanism available – at the time of writing this article – that can traverse across multiple RDF triple stores in a federated manner to retrieve properties (or paths) that connect two given nodes. Manually identifying which of the LOD datasets contain the connecting properties is impractical, cumbersome, and a time-consuming process. For instance, in the current scenario, a SPARQL 1.1 property path query (shown in the Listing 1.1)

Listing 1.1. Property Path query in SPARQL1.1

```
Prefix : <urn:exe:>
SELECT ?s ?p ?o
{ :cnv (:|!:) * ?s . ?s ?p ?o . ?o (:|!:) * :gene }
Group By ?s ?p ?o
```

would return all the *genes* which *:cnv* (*copy number variation*) connects to via any property in a single RDF graph. Although the SPARQL 1.1 version supports federation (via the SERVICE operator) and property path features, it is surprising that existing approaches and open source triple stores like Virtuoso and Jena does not provide any solution to federate path queries over distributed graphs. Similarly, there is numerous research already carried out in traversing and/or finding shortest paths in a graph [6,13]. In the context of Linked Data, the European Semantic Web Conference (ESWC) in 2016 hosted a challenge [15] to find “Top-K Shortest Path in Large Typed RDF Graphs” where the main focus of all the competing participants has been on the extending or optimising the state-of-the-arts graph traversal algorithm over single RDF graph; none were aimed to navigate across federated graphs. Therefore, to find the path between source and target, the centralised approaches adopted by current systems pose some challenges such as; (i) user must know the priori knowledge of underlying schema of data to query on, (ii) requires to merge whole data into a single graph, which is a cumbersome task, (iii) copied data needs to be synchronized, and (iv) it lacks the opportunity to query the up-to-date and fresh data. while on the other-side federation provides the opportunity to tackle these challenges. Hence, to address these challenges we propose FedS that takes as input source (subject), target (object) nodes, and a list of triple stores to federate upon; and returns a list of paths that connect the source and target nodes across the federated triple stores.

In our previous work [17], we proposed an extension of SPARQL which allows finding the top-k shortest paths on a single compressed RDF graph – in the

² <http://bio2rdf.org/>.

³ <http://linkedlifedata.com/>.

HDT [3] format – using the property path expression. In this paper, we propose FedS that federates across multiple triple stores by (i) selecting prospective triple stores where source and target nodes are available; (ii) retrieving paths from different triple stores; and finally (iii) merging all the retrieved paths that connect source and target nodes hosted in different triple stores. We start the paper by presenting a motivation scenario that aims to retrieve paths between source and target nodes hosted in different triple stores. We then discuss the related works of traversing and retrieving paths in graph databases. We then present the FedS architecture and description of its four core components. We provide an evaluation of FedS compared to the state-of-art approaches. Finally, we present our conclusion and various routes to optimise the navigation across federated RDF graphs.

2 Motivation Scenario - Cancer Genomics

In order to understand the cancer progression, it is often the case that several genetic features, diseases, medical history, etc. are studied together, therefore, one of the key challenge in cancer genomics – a cornerstone of precision medicine – is to discover gene-disease-drug associations. Such novel associations provide insight into the drug development process tailored specifically for an individual patient (or a group of patients) targeting prevention, diagnosis and treatment of the diseases [18].

For instance, consider a scenario where a biomedical expert is trying to discover the paths between a drug (*drugbank* : *DB00222*) and a drug compound (*kegg* : *C07669*). The Fig. 1 shows a group of datasets (DrugBank, KEGG, Hgnc, OMIM, and Pharmgkb) hosted at five different SPARQL endpoints. The source drug (*drugbank* : *DB00222*) is located in the SPARQL endpoint for DrugBank, whereas the target drug compound (*kegg* : *C07669*) exists in the SPARQL endpoints 1 (DrugBank), 2 (KEGG), and 5 (Pharmgkb). The source (*drugbank* : *DB00222*) – Glimepiride – is an antidiabetic drug whereas the target (*kegg* : *C07669*) represents a drug compound associated with Glimepiride.

If the biomedical expert needs to establish associations between the drug and its compound, he/she can find the direct correlation by querying the SPARQL endpoint for DrugBank. However, for further analysis to understand the mechanism of a drug compound and affected biological pathways, the expert needs to explore and discover associations in various others biomedical datasets. Figure 1 shows three paths starting from the *drugbank* : *DB00222* (SPARQL endpoint 1) where the target (*kegg* : *C07669*) is available in (i) the DrugBank endpoint itself via the *:x-kegg* property; (ii) the Kegg endpoint via the *:x-kegg* and *:sameAs* properties; and (iii) the Pharmgkb endpoint via the *:x-pharmgkb* and *:x-kegg* properties. The fourth path at the HGNC endpoint contains a node (*hgnc* : *2623*) via the *:enzyme* and *:x-hgnc* properties, however, the target node (*kegg* : *C07669*) doesn't exist in this endpoint.

We believe that a technology that can enable querying paths/associations among two or more biological entities across distributed repositories would be a

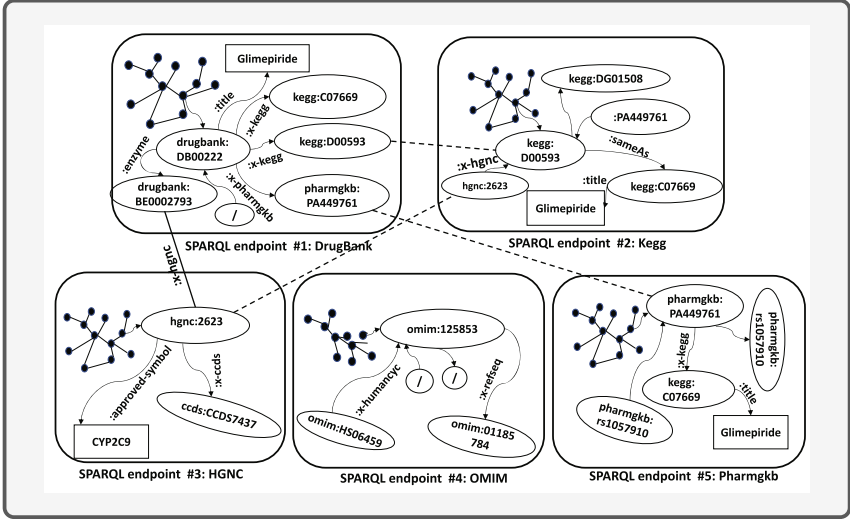


Fig. 1. Navigation Across Federated SPARQL endpoints

great help to biologist and practitioners working in the cancer genomics as well as in the larger healthcare and life sciences area.

3 Preliminaries

Consider a network of distributed datasets where each dataset loaded with $G = (V, P)$ directed graph. V is the set of vertices and P is the set of paths (edges). The vertices (v_i, v_j) are associated through a set of paths (edges) p_{ij} . If there exists a path between (v_i, v_j) then v_j is the *successor* of v_i and v_i is the predecessor of v_j . For RDF graph, we define as follows:

Definition 1 (RDF Triple & Graph). *The set of RDF terminologies consists of the set of IRIs I , the set of blank-nodes B and the set of literals L . An RDF Triple $T := (s; p; o)$ is an element of the set $G := (I \cup B) \times I \times (I \cup L \cup B)$. The set G is a finite set of triples called RDF graph. A RDF graph G_i of a single triple T_i represents a vertex (subject) s_i associated with another vertex (object) o_i via a path (property) p_i .*

In a federated environment, if a path between source and target vertices does not exist within a local single graph, other remote graphs are scanned and queried to find the path existence between given vertices over the network.

Definition 2 (FedS Source Selection). *In a federated query environment, given a triple pattern T with subject s_i and object o_i vertices in a bgp in a query Q executed against data sources \mathcal{D} , the set of relevant sources for T in \mathcal{D} is the set $\mathcal{R}_t \subseteq \mathcal{D}$ of data sources that can provide answers when queried with T .*

We use the notation \mathcal{R}_T to denote the set of relevant data sources for source (subject) and target (object) nodes and use \mathcal{R} when the context does not require specifying the triple patterns.

FedS performs a source selection process where a set of relevant data sources for a given query are discovered by scanning all the given data sets \mathcal{D} and input triple pattern T .

Definition 3 (FedS Reachability). *In case of a labeled directed RDF graph G , the reachability relation is the transitive closure of RDF properties $p(s,o)$ such that for the set of all ordered paired of subjects (s) and objects (o) there exists a sequence of subjects and objects $s = v_o, v_i, \dots, k = o$ where the property $p(v_{i-1}, v_i)$ is in $p(s,o)$ for all $1 \leq i \leq k$.*

For a given query to find the path between (s,o), if these two are connected through any number of paths p_i, \dots, n we say that path exists and source(s) and target(o) are reachable. While if there is no path between (s,o) it means that (s,o) is not reachable.

4 Related Work

Numerous amount of work has been done on the graph navigation and pathfinding problems [13] and variety of algorithms have been proposed to compute the shortest path within the given networks. Similarly, there is an extensive set of works [4,6,7,14] related to the performance of pathfinding algorithms in very large graphs. They employ different techniques to get the optimised performance, for instance, in case of [4,7], the algorithms only work on compressed datasets stored in customised databases (e.g., HDT, RDF-3X), while others have worked on the graph partitioning problem. Our previous work [17] along with others [8,10,11,16] proposed extensions for SPARQL 1.1 property paths, but none of them consider multiple triple stores during the graph navigation process. Therefore, we have no work to compare that does graph traversal over federated triple stores. Thus the motivation for our research is to investigate, how to enable path evaluation over federated triple stores. To the best of our knowledge, FedS is the first attempt to retrieve (evaluate) paths across the graphs hosted in federated triple stores.

5 FedS

FedS approach works as peer-to-peer network of triple stores where every triple store has FedS running on it. FedS extends previous work [17] to federate path traversal across the network. The FedS architecture is summarised in Fig. 2, which shows its four core components: (i) **Source Selection**: performs source selection based on the availability of a source node within triple stores; (ii) **Path Computation**: once a subset of triple stores are identified that host the source node,

path traversal starts from one of the selected triple stores; (iii) **Path Federation**: when the path traversal starts at host triple store, the FedS probes other RDF stores – during the iteration – wherever the target node is available; and finally (iv) **Path Merger**: it aggregates all the path retrieved from different triple stores and prepare a list of paths between the source and target nodes.

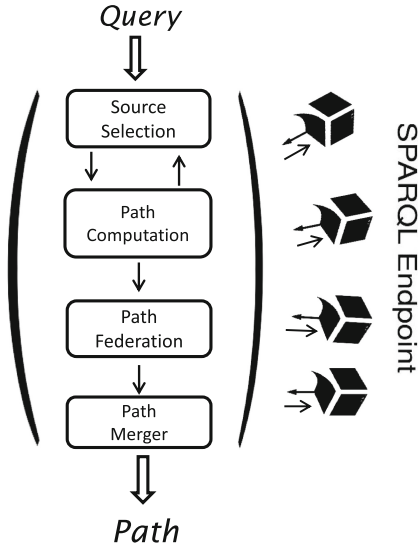


Fig. 2. FedS architecture

Source Selection: FedS performs a partially index free source selection of relevant triple stores (see Definition 2), where only the address IRIs are stored. The source selection is performed by using the SPARQL ASK query that returns the availability of a source and target in different triple stores. For example, given the list of five RDF stores (DrugBank, HGNC, PharmGKB, OMIM, and KEGG) (see Fig. 1), the source selection process identifies three triple stores (DrugBank, KEGG, Pharmgkb) where the source (*drugbank* : *DB00222*) and target (*keg* : *C07669*) nodes exist. Then, one of the identified triple store – where the source node is found – is randomly selected to initialize the path traversal process.

Path Computation: The traversal starts from the identified triple store that contains the source node and navigates along all the paths (*successors*) and check the reachability between source and target nodes (see Definition 3). If the target node is found in the hosted triple store, it feeds (reports) the retrieved path to the “Path Merger” component. However, if path from source to target node is not reachable – within the hosted triple store – we defined it as a *partial path*. The partial paths are created from the source to all its successors up to the leaf (last) nodes in the current triple store. For instance, two simple partial paths;

kegg:D00593 and *pharmgkb:PA449761* are one hop away from source *DB00222*, therefore are the leaf nodes. However, there always be complex partial paths where many nodes exist from root to last node (source) to its last node. FedS considers all the nodes of partial paths to check their availability in other datasets and how this happens is explained in “Path Federation” component. Further, there may also exist indirect paths between source and target. For instance, in Fig. 1 endpoint 1 is linked with endpoint 3 through $DB00222 \rightarrow \text{:anzyme} \rightarrow BE0002793 \rightarrow \text{x:hgnc} \rightarrow \text{hgnc:2623}$. Endpoint 3 does not contain the target node *C07669*, however, *hgnc:2623* is connected to endpoint 2 where actual target node *C07669* does exist. To compute the indirect paths, FedS performs the following steps; (i) partial path is created $DB00222 \rightarrow \text{:anzyme} \rightarrow BE0002793 \rightarrow \text{x:hgnc} \rightarrow \text{hgnc:2623}$, (ii) *hgnc:2623* is not target node therefore FedS ASK this node and finds that endpoints 2 has *hgnc:2623*. From endpoint 2 partial path – *hgnc:2623* $\rightarrow \text{:xhgnc} \rightarrow D00593 \rightarrow \text{:sameAs} \rightarrow C07669$ – is returned and is reported to the component “Path Merger”.

Path Federation: This component iterates over the list of partial paths where each element (node) involved in partial path along with actual target node is checked (using the SPARQL ASK construct) for their availability in the already identified triple stores. Upon receiving the federated request, the traversal process starts in all those triple stores which contain the corresponding nodes and returns the retrieved paths towards the endpoint from where the request was dispatched. For example, the two dashed lines between **DrugBank-Kegg** and **DrugBank-Pharmgkb** endpoints shows the availability of nodes (*kegg : D00593* and *pharmgkb : PA449761*) in these two endpoints.

Path Merger: This component stores and merge all the partial paths and the paths retrieved from different SPARQL endpoints. When the paths are received from different SPARQL endpoints, this component concatenates each retrieved path to the corresponding node in the partial list such that a complete path is completed (i.e., partial from local and partial from remote enapoint) paths from source to target nodes. For example, two direct complete paths are: (i) $DB00222 \rightarrow \text{:x-pharmgkb} \rightarrow PA449761 \rightarrow \text{:x-kegg} \rightarrow C07669$; and (ii) $DB00222 \rightarrow \text{:x-kegg} \rightarrow D00593 \rightarrow \text{:sameAs} \rightarrow C07669$). There also exists an indirect path which is; $[DB00222 \rightarrow \text{:anzyme} \rightarrow BE0002793 \rightarrow \text{x:hgnc} \rightarrow \text{hgnc:2623} \rightarrow \text{x:hgnc} \rightarrow D00593 \rightarrow \text{:sameAs} \rightarrow C07669]$

The main objective of FedS is to retrieve paths over the federated network of triple stores and the major advantages are: (i) FedS doesn’t need priori knowledge of underlying schema of RDF graphs hosted in different SPARQL endpoints; (ii) In order to traverse the paths, FedS does not require to merge the federated RDF graphs into a single graph; and (iii) finally, FedS is still be able to compete with state-of-the-art approaches.

6 Results and Discussion

The experimental setup comprises of datasets (i.e., SPARQL endpoints) and six input queries. we compared Feds against three systems (Virtuoso, Blazegraph, and HDT-Bidirectional-TopK).

Datasets: We have downloaded the release-4 of Bio2RDF datasets⁴. The data cumulatively is around 3.89 GB – DrugBank (size 1.18 GB), KEGG (size 471.9 MB), PharmaGKB (size 29.2 MB), OMIM (size 1.61 GB), and HGNC (size 592 MB) – with 22,36,32,57 triples, 2,343,770 subjects, 334 predicates and 81,599,44 objects. Table 1 shows the number of triples, subjects, predicates, and objects in each dataset and the hardware specs of five desktop machines used to conduct the experiments are shown in Table 2.

Table 1. Datasets statistics

Dataset	Size	Triples	Subject	Predicates	Objects
Drugbank	1.18 GB	5151714	421348	104	2472011
Kegg	479.1 MB	3281579	358844	63	1835508
Pharmgkb	29.2 MB	191379	19905	32	123336
Omim	1.61 GB	9687186	1127394	93	1415364
HGNC	592 MB	4051399	416279	42	2313725

Table 2. Specifications of virtual machines used in experiments

OS	Data loaded	RAM	Hard disk	Processor
MAC	Omim	16 GB	500 GB	2.6 GHz Intel Core i5
Ubuntu	Drugbank	32 GB	500 GB	2.9 GHz Intel Core i7
Windows	Kegg	8 GB	250 GB	2.2 GHz Intel Core i7
Ubuntu	Hgnc	8 GB	300 GB	2.5 GHz Intel Core i5
Windows	Pharmgkb	16 GB	250 GB	2.2 GHz Intel Core i5

Queries: The Table 3 shows six evaluation queries. The column two in Table 3, shows the number of datasets (SPARQL endpoints) selected for each query. For example, in the case of **Q1** DrugBank and Pharmgkb are needed to retrieve the complete paths. Similarly, **Q5** needed 3 SPARQL endpoints (DrugBank, Kegg, and Pharmgkb) to retrieve all three paths. The “Hops” column denote the number of properties (named edges) between source and target nodes.

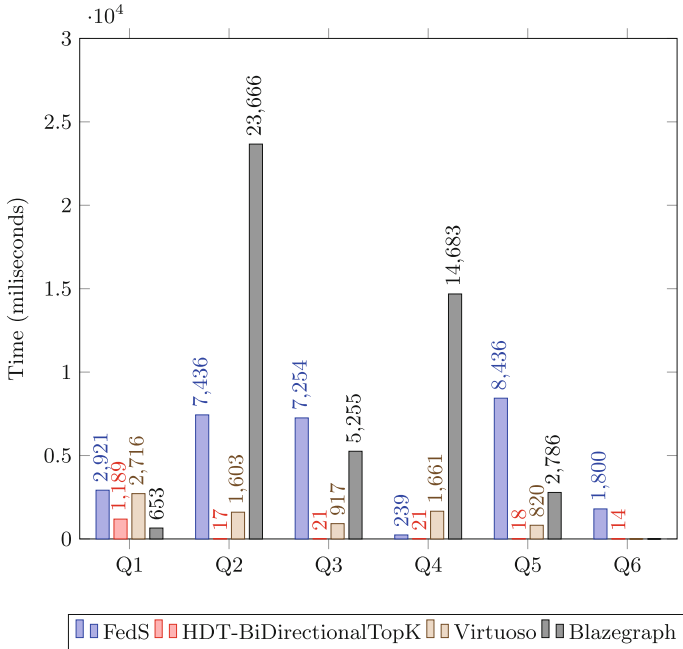
Performance Analysis: In order to compare against the triple stores (Virtuoso, Blazegraph, and HDT-BidirectionalTopK) which support the SPARQL

⁴ <http://download.openbiocloud.org/release/4/>.

Table 3. Six Input Queries and Results

Query	Dataset selected	Source	Target	Paths	Hops
Q1	Drugbank — Pharmgkb	drugbank:DB00072	clinicaltrials:NCT01959490	1	2
Q2	Drugbank—HGNC	drugbank:DB01268	hgnc.symbol:FLT3	1	3
Q3	Drugbank	drugbank:DB00134	kegg:D04983	1	2
	Drugbank—Kegg	drugbank:DB00134	kegg:D04983	1	1
Q4	Drugbank	drugbank:BE0002362	hgnc.symbol:CYP3A5	1	1
Q5	Drugbank	drugbank:DB00222	kegg:C07669	1	1
	Drugbank—Kegg	drugbank:DB00222	kegg:C07669	1	2
	Drugbank—Pharmgkb	drugbank:DB00222	kegg:C07669	1	2
	Drugbank—HGNC—Kegg	drugbank:DB00222	kegg:C07669	1	4
Q6	Drugbank	omim:147470	hgnc.symbol:IGF2	3	3x1

1.1 Property Path specification. We performed following steps: (i) merged all the data within five SPARQL endpoints into a single graph and loaded it into each triple store (*i.e.*, Virtuoso, Blazegraph, HDT-BidirectionalTopK); and (ii) executed SPARQL 1.1 property path query without specifying any predicate in path expression (e.g., (:|:)*) on each triple store. As shown in the Fig. 3, HDT-Bidirectional-TopK outperforms all other systems simply because it works only on HDT-based compressed RDF data rather than raw RDF data.

**Fig. 3.** Comparing total query execution time

FedS, Virtuoso, and Blazegraphs work only on raw RDF data and therefore results are comparable. The key point to be noted here is: FedS has to cope with additional messaging over the network (*e.g.* network cost due to SPARQL ASK queries, machines with different specs, source selection, federation request, etc.). We also noticed that Virtuoso and Blazegraph were not able to return the results for **Q6**. In case of Virtuoso, all queries were showing *time-out* if the queries were executed without specifying the named graphs⁵.

Path Analysis: Table 4 shows the comparison of *total number of paths* retrieved and their corresponding *path length* (*i.e.*, Hops). In terms of result completeness – as triple stores are designed to retrieve complete sets of path in a single graph – for all the six queries, FedS retrieved the equal number of paths in a federated environment.

Table 4. Comparison of the Total Paths **#TP** and Path Hops **#PH**

Query	FedS		HDT-BiDirectionalTopK		Virtuoso		Blazegraph	
	#TP	#PH	#TP	#PH	#TP	#PH	#TP	#PH
Q1	1	2	1	2	1	2	1	2
Q2	1	3	1	3	1	3	1	3
Q3	1	2		2	1	2	1	2
	1	1	1	1	1	1	1	1
Q4	1	2	1	2	1	2	1	2
Q5	1	2	1	2	1	2	1	2
	1	2	1	2	1	2	1	2
	1	1	1	1	1	1	1	1
	1	4	1	4	1	4	1	4
Q6	3	3x1	3	3x1	-	-	-	-

7 Conclusion and Future Work

In this paper we propose FedS, a path traversal approach that federates across multiple SPARQL endpoints. This work is motivated by the needs of biomedical domain to find associations (paths) across different biological entities. The current SPARQL 1.1 Property Path specification and the standard traversal algorithms (BFS, DFS, A*, etc.) assume (or require) a single graph – or many graphs merged into a centralised graph – for graph traversal. FedS proposes a four step process that enables graph traversal in a federated environment. Our initial evaluation results are encouraging where FedS retrieves all the paths in a competing query processing time when compared to state-of-the-art triple stores approaches. In terms of future work, there are a number of possible routes to

⁵ <https://www.w3.org/TR/rdf-sparql-query/#namedGraphs>.

optimise the current four step process (i) we plan to implement a ranking mechanism in the source selection process to select the *top-k* SPARQL endpoints to initialise the path navigation process; (ii) next obvious step is to devise a method that ranks (*top-k*) the retrieved paths between source and target nodes; (iii) we plan to include larger data sets with significant number of paths (i.e., 10+) between source and target nodes; and finally, (iv) the current implementation navigates only in the forward direction, to further optimise, we plan to implement navigation in both the directions.

Acknowledgements. The work presented in this research paper has been funded by Science Foundation Ireland under Grant No. SFI/12/RC/2289.

References

1. Bio2RDF Release 3: A larger, more connected network of Linked Data for the Life Sciences, vol. 1272. CEUR Workshop Proceedings, Riva del Garda, Italy. CEUR-WS.org (2014)
2. Belleau, F., et al.: Bio2rdf: towards a mashup to build bioinformatics knowledge systems. *J. Biomed. Inf.* **41**(5), 706–716 (2008)
3. Fernández, J.D., Martínez-Prieto, M.A., Gutiérrez, C., Polleres, A., Arias, M.: Binary RDF representation for publication and exchange (HDT). *J. Web Sem.* **19**, 22–41 (2013)
4. Filtz, E., Savenkov, V., Umbrich, J.: On finding the k shortest paths in RDF data. In: 5th International Workshop (IESD 2016) co-located with the (ISWC 2016), vol. 18 (2016)
5. Gao, J., et al.: Integrative analysis of complex cancer genomics and clinical profiles using the cbiportal. *Sci. Signal.* **6**(269), p11 (2013)
6. Goldberg, A.V.: Point-to-point shortest path algorithms with preprocessing. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) SOFSEM 2007. LNCS, vol. 4362, pp. 88–102. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-69507-3_6
7. Gubichev, A., et al.: Fast and accurate estimation of shortest paths in large graphs. In: Proceedings of the 19th ACM CIKM. ACM (2010)
8. Gubichev, A., Neumann, T.: Path query processing on very large RDF graphs. In: WebDB, Citeseer (2011)
9. Hu, W., Qiu, H., Dumontier, M.: Link analysis of life science linked data. In: Arenas, M., et al. (eds.) ISWC 2015. LNCS, vol. 9367, pp. 446–462. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25010-6_29
10. Kochut, K.J., Janik, M.: SPARQLeR: extended SPARQL for semantic association discovery. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 145–159. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72667-8_12
11. Kostylev, E.V., Reutter, J.L., Romero, M., Vrgoč, D.: SPARQL with property paths. In: Arenas, M., et al. (eds.) ISWC 2015. LNCS, vol. 9366, pp. 3–18. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25007-6_1
12. Lacroix, Z., Murthy, H., Naumann, F., Raschid, L.: Links and paths through life sciences data sources. In: Rahm, E. (ed.) DILS 2004. LNCS, vol. 2994, pp. 203–211. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24745-6_14

13. Madkour, A., Aref, W.G., Ur Rehman, F., Rahman, M.A., Basalamah, S.: A survey of shortest-path algorithms. arXiv preprint [arXiv:1705.02044](https://arxiv.org/abs/1705.02044) (2017)
14. Möhring, R.H., Schilling, H., Schütz, B., Wagner, D., Willhalm, T.: Partitioning graphs to speedup dijkstra's algorithm. *J. Exp. Algorithmics (JEA)* **11**, 2–8 (2007)
15. Papadakis, I., Stefanidakis, M., Mylonas, P., Niggemeyer, B.E., Kazanas, S.: Top-K shortest paths in large typed RDF datasets challenge. In: Sack, H., Dietze, S., Tordai, A., Lange, C. (eds.) *SemWebEval 2016*. CCIS, vol. 641, pp. 191–199. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46565-4_15
16. Przyjaciel-Zablocki, M., Schätzle, A., Hornung, T., Lausen, G.: RDFPath: path query processing on large RDF graphs with MapReduce. In: García-Castro, R., Fensel, D., Antoniou, G. (eds.) *ESWC 2011*. LNCS, vol. 7117, pp. 50–64. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-25953-1_5
17. Savenkov, V., Mehmood, Q., Umbrich, J., Polleres, A.: Counting to k, or how sparql1.1 property paths can be extended to top-k path queries. In: *SEMANTICS 2017* (2017)
18. Simon, R., Roychowdhury, S.: Implementing personalized cancer genomics in clinical trials. *Nat. Rev. Drug Dis.* **12**(5), 358–369 (2013)

Case Studies



Adversarial Spiral Learning Approach to Strain Analysis for Bridge Damage Detection

Takaya Kawakatsu¹(✉), Akira Kinoshita², Kenro Aihara², Atsuhiko Takasu²,
and Jun Adachi²

¹ The University of Tokyo, 2-1-2 Hitotsubashi, Chiyoda, Tokyo, Japan
kat@nii.ac.jp

² National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda, Tokyo, Japan

Abstract. When a vehicle passes over a bridge, the bridge distorts in response to the vehicle's load. The response characteristics may change over time if the bridge suffers damage. We consider the detection of such anomalous responses, using data from both traffic-surveillance cameras and strain sensors. The camera data are utilized to treat each vehicle's identified properties as explanatory variables in the response model. The video and strain data are transformed into a common feature space, to enable direct comparisons. This space is obtained via our proposed spiral learning method, which is based on a deep convolutional neural network. We treat the distance between the video and strain data in the space as the anomaly score. We also propose an adversarial unsupervised learning technique for removing the influence of the weather. In our experiments, we found anomalous strain responses from a real bridge, and were able to classify them into three major patterns. The results demonstrate the effectiveness of our approach to bridge damage analysis.

Keywords: Structural health monitoring · Deep learning
Multimedia analysis

1 Introduction

Many road bridges built in Japan in the 1960s have deteriorated and now require substantial inspection. We approach the health monitoring problem by installing a number of inexpensive sensors on a bridge as shown in Fig. 1. We aim to identify small signs of bridge deterioration by developing a fully data-driven inspection technique. Our research started from a simple question. Could the bridge strain response caused by a vehicle be predicted from the surveillance video, using an *encoder-decoder* [1] approach? If we set up such a predictor during the bridge's construction, we could capture small signs of deterioration later by monitoring the frequency of *anomalous* vehicles. Unfortunately, we encountered a difficulty, namely that the video data could not supply information about a vehicle's axle weights.

In general, the strain response will depend on the moving axle loads, so measuring axle weights directly seems an obvious approach. To obtain the axle weights, the options are to install a pavement sensor or use a bridge weighing in motion (BWIM) technology [2]. The former is fragile, hard to retrofit to existing bridges, and limits the traveling speeds. The latter requires that strain response characteristics be obtained in advance, which is not applicable to the anomaly detection problem. We, therefore, abandoned the axle weight approach.

Instead, we developed a new sensor fusion approach that directly compares the vehicle image and strain response in a common feature space. This approach is somewhat similar to the Siamese network [3], except that our approach utilizes two different neural networks for the video and strain data. Moreover, unlike the Siamese network, the two networks are trained to predict vehicle speeds and loci individually. As we reported in a previous paper [4], vehicle speeds and loci may be predictable from both video and strain data, and they affect the shape of the strain response significantly. By learning these two tasks, the two networks can acquire feature spaces that seem to comprise *common factors* of the video and strain data. Finally, the bases of the two spaces can be matched by minimizing the distance between two related elements in the respective spaces. We call this approach *spiral learning*. Some video–response pairs caused by the same vehicle may be inconsistent in the common space. We treat such an event as an *anomaly*, making the assumption that such a response may be caused by bridge structural damage. It should be noted that bad weather, e.g., a snowstorm or heavy rain, may disrupt the video signal. In such a situation, an event may be misidentified as an anomaly, even if the strain response was normal. Therefore, we proposed adding an *adversarial learning mechanism* as a countermeasure.

We evaluated our proposed approach using real observations recorded over a six-month period. We then classified the observed anomalies into some patterns to demonstrate the weather resistance. The results show the effectiveness of our approach to bridge damage analysis.

2 Bridge Analysis: The Influence Line

Studies of bridge damage detection can be classified into two approaches, namely steady-state analysis [5] and transient-response analysis [6, 7]. The former detects damage by detecting changes in natural frequencies and time constants, whereas the latter aims to detect damage by focusing on temporary events such as passing vehicles. These approaches require the vehicle speed, locus (traveling position in the lane), and axle positions, as explanatory variables in the response model.

A bridge bends as vehicles pass over it. If we assume a linear-response model, the strain is proportional to the vehicle weight and closeness between the sensor and the vehicle. The bending moment $m(t)$ at time t will vary depending on the vehicle position, x . $m(t)$ can be estimated by the *influence line* $i(x)$ and strain measurements $s(x, t)$ given in Eq. (1):

$$\hat{m}(t) = \int_0^l w(x, t)i(x)dx \approx \int_0^l ES(x)s(x, t)dx, \quad (1)$$

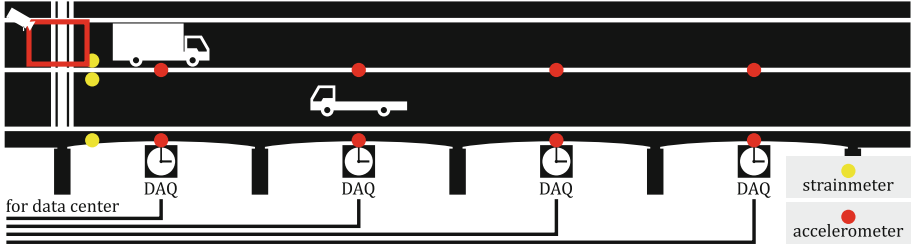


Fig. 1. Sensor positions.

where $\hat{m}(t)$ is the predicted moment and $w(x, t)$ is the axle weight at x . E is the elasticity modulus and $S(x)$ is the section modulus.

The influence line characterizes an individual bridge but may change its shape over time owing to structural damage to the bridge. A number of researchers [6, 7] evaluated a damage detection technique using influence lines. To obtain an ideal influence line, we should run a test vehicle with known axle weights, speed, and locus, excluding the influence of other vehicles. However, that can prove difficult, particularly for bridges carrying heavy traffic.

Zaurin and Catbas [8] proposed a novel sensor fusion approach for monitoring bridges in service. They installed a traffic-surveillance camera and strain sensors on a miniature bridge. The camera was used for axle detection. The axle positions were obtained by tracking axles and were used for calculating the surface load $w(x, t)$. The authors [8] assumed that axle weights could be measured via BWIM. The influence line was estimated for every vehicle. By clustering the estimated influence lines, the potential damage was identified. Their approach has a serious weakness in that BWIM requires obtaining the influence line in advance. Once the bridge becomes degraded, the influence line may change, which means that the BWIM system may be unreliable for their purpose.

3 The Spiral Learning Approach

Spiral learning is a technique whereby a pair of samples from two data sources are fed into two separate networks. The networks are independent of each other, except for a final linear layer that shares the same weight matrix. Each network learns from its respective dataset and acquires a feature mapping that contains information about vehicle properties such as speed and locus. The samples refer to the same vehicle as a latent variable.

Table 1 shows the neural network architectures. $[k/d, c]$ denotes a c -channel convolution layer with a kernel width of k , which downsizes the sample to $1/d$ -th of the original. Each stacked matrix denotes a plain residual block [9]. CamNet receives 50 grayscale video frames (taken over two seconds), which are resized to 224×224 pixels, and outputs the estimated speed and locus of the target vehicle. SigNet receives four-second batches of raw strain data sampled at 200 Hz. Each strain sample was rescaled so that its maximum and minimum were normalized to

Table 1. Network architectures for video and strain data.

Layers	CamNet	SigNet
Conv 1	$3/2, 50$ $3/2, 50$	$[25/4, 64]$
Conv 2	$3/1, 50$ $3/1, 50$	$7/1, 64$ $7/1, 64$
Conv 3	$3/2, 50$ $3/2, 50$	$7/4, 64$ $7/4, 64$
Conv 4	$3/1, 50$ $3/1, 50$	$3/1, 64$ $3/1, 64$
Conv 5	$3/2, 50$ $3/2, 50$	$3/1, 64$ $3/1, 64$
Conv 6	$3/1, 50$ $3/1, 50$	$3/4, 64$ $3/4, 64$
Conv 7	$3/2, 50$ $3/2, 50$	$3/1, 64$ $3/1, 64$
Conv 8,9	$3/1, 50$ $3/1, 50$	$3/1, 64$ $3/1, 64$
Linear 1	Output: 100×1	
Linear 2	Output: Speed and locus	

1 and 0, respectively for effective learning. We used ReLU [10] for the activation functions in each layer of the two networks, except for the output layers.

We selected speed and locus as the outputs because these two properties have the next strongest effect on the signal shape of the strain response, after the axle loads. We can expect the two networks to generate the same feature vector as a common factor, through learning the two prediction tasks and sharing the same output layer. The loss function for CamNet is defined in Eq. (2):

$$\mathcal{L}_{\text{MSE1}}(f, h) = \frac{1}{N} \left\{ \sum_{n=1}^N \frac{[h_s(f(\mathbf{x}_n)) - s_n]^2}{\text{Var}(s)} + \sum_{n=1}^N \frac{[h_l(f(\mathbf{x}_n)) - l_n]^2}{\text{Var}(l)} \right\}, \quad (2)$$

where \mathbf{x}_n is the n -th video sample. f and h denote the feature extraction through CamNet and the Linear 2 layer, respectively. s and l are ground truth annotations for the speed and locus prediction tasks. The Linear 2 layer is shared by the two models, enabling the outputs of the Linear 1 layer to be treated as feature vectors in a common feature space. Both networks learn the correlation between the two sources by drawing their feature vectors together. The attracting mechanism can be described as the anomaly loss $\mathcal{L}_{\text{MSE3}}$ defined in Eq. (3).

$$\mathcal{L}_{\text{MSE3}}(f, g) = \frac{\lambda}{N} \sum_{n=1}^N \left\| \frac{f(\mathbf{x}_n)}{\|f(\mathbf{x}_n)\|_2} - \frac{g(\mathbf{y}_n)}{\|g(\mathbf{y}_n)\|_2} \right\|_2^2, \quad (3)$$

where λ is a weight of $\mathcal{L}_{\text{MSE3}}$ and is set as 10. g denotes the feature extraction through SigNet and \mathbf{y}_n is the n -th strain sample. Consequently, the optimization problem for the combined network, named SpiNet, can be described in terms of multitasking [11], as given in Eq. (4).

$$\mathcal{L}_{\text{TMSE}}(f, g, h) = \mathcal{L}_{\text{MSE1}}(f, h) + \mathcal{L}_{\text{MSE2}}(g, h) + \mathcal{L}_{\text{MSE3}}(f, g). \quad (4)$$

Equation (4) minimizes five individual losses, namely speed and locus prediction from the video data, speed and locus prediction from the strain data, and the L^2 norm between the feature vectors in CamNet and SigNet. Because these two networks share the output layer, Eq. (3) plays the role of matching the correlative elements from the two feature spaces. As a result, the video and strain feature spaces will coalesce after a long training period, as two *cannibal black holes* forming a spiral trajectory. The video and strain data may *differ* from each other, even though they cover the same target vehicle. This can be identified by monitoring the L^2 norm of the subtraction between $f(\mathbf{x})$ and $g(\mathbf{y})$. We, therefore, define $\mathcal{L}_{\text{MSE3}}$ as an *anomaly score*, setting λ as 1. The outliers of the distribution will be identified as *anomalous* vehicles.

The anomalous vehicle detection based on spiral learning depends on features extracted from the surveillance video recordings. The video can be disturbed by environmental conditions such as weather, traffic jams, pedestrians, and vehicles in the opposite lane. Such disturbances may cause mistaken anomaly detections. The *adversarial spiral learning* can reduce these errors, particularly those caused by bad weather. It may enable video features to correlate less with the weather conditions, including heavy rain, snowstorms, deep snow, and morning haze.

We, therefore, combined the adversarial learning concept [12] with our spiral learning proposal for this purpose. Adversarial learning has been introduced for image generation, utilizing a discriminator and a generator implemented by two separate neural networks. The discriminator finds fake images from a given image set that includes real and generated images. The generator creates images that are exactly like the real ones, which the discriminator may then misjudge as real images. The training mechanism can be described as a discriminator aiming to minimize the discrimination error whereas the generator aims to maximize it.

One of the simplest methods to achieve weather resistance is to use a weather discriminator. The discriminator tries to find videos recorded under bad weather conditions by examining video features carefully and in detail. To implement this function, we need to append weather tags to the traffic dataset. The loss function for SpiNet can be described as in Eq. (5), using the mean cross entropy \mathcal{L}_{MCE} :

$$\mathcal{L}_{\text{spin}}(f, g, h) = \mathcal{L}_{\text{TMSE}}(f, g, h) - \mathcal{L}_{\text{MCE}}(p, q), \quad (5)$$

where p and q denote the discriminator and the weather tag, respectively. After a long training period, the discriminator can no longer find faults in the obtained video features.

Initially, we tried to tag each vehicle by consulting the historical climate data archived by the government, but we encountered a major difficulty. The weather conditions at the bridge did not always correspond with the historical data.

This was because weather conditions were recorded at the nearest observation station, a few kilometers away from the target bridge. We then tried to tag the vehicles manually by watching the video, but encountered another difficulty. Because of the complicated weather situations, it was hard to formulate a robust policy for weather annotation. For example, should there be a cloudy tag in addition to a sunny tag, and is there a boundary between cloudy and light-rain conditions, or between snowfall and snow accumulation? Sometimes, the video lost focus owing to morning haze, twilight, or a fogged lens. Additionally, not only precipitation but also the intensity of solar radiation, which also has a strong impact on the video quality, should be noted.

We, therefore, abandoned the weather-annotation plan and developed a fully unsupervised approach. Here, the issue was that a vehicle might be mistakenly judged anomalous because of bad weather, even though the strain response was normal. This was a situation where an *anomalous strain response* could be found without consulting the strain feature, but by consulting the video feature alone instead. This might be a problematic situation, considering the main purpose of the strain characteristics analysis. We therefore defined the ground truth tag for the discriminator as given in Eq. (6):

$$q(\mathbf{x}, \mathbf{y}, f, g) = \mathbb{H} \left\{ \|f(\mathbf{x}) - g(\mathbf{y})\|_2^2 - \mathcal{L}_{\text{MSE3}}^{\text{train}}(f, g) \right\}, \quad (6)$$

where \mathbb{H} is the step function and $\mathcal{L}_{\text{MSE3}}^{\text{train}}$ is the anomaly loss (3) for the training dataset. The initial value of $\mathcal{L}_{\text{MSE3}}^{\text{train}}$ for the first epoch was 0.1. The adversarial network, named TwiNet, was defined as a perceptron whose hidden layer has 10 dimensions.

4 Training and Evaluation Data

We conducted our experiments on a 300-m-long prestressed concrete bridge in Japan. The bridge had four spans and two lanes as shown in Fig. 1, and has suffered damage caused by snowy weather. We have deployed a highly sensitive strainmeter and a surveillance camera on the bridge. The strain sensor observed the horizontal strain of its deck slab, sampled at 200 Hz.

To make a traffic dataset for the experiments in Sect. 5, we improved the traffic-surveillance system (TSS) [13]. After a vehicle enters the bridge, a camera installed at the entrance captures the vehicle. TSS detects the bounding box for the vehicle image, by using Faster R-CNN [14]. TSS outputs data about vehicles one by one, including properties such as lane, speed and locus. The mechanism for estimating vehicle speeds and loci was as follows. First, the TSS identifies all possible pairs of two axles between two consecutive video frames. Next, the TSS calculates the amount of movement for all pairs. Finally, the TSS estimates the traveling speed from the median of the movement amounts, and the locus from the average bottom position of the frontmost axles. It should be noted that the coordinates are transformed so that the distance in pixels is proportional to the distance in meters.

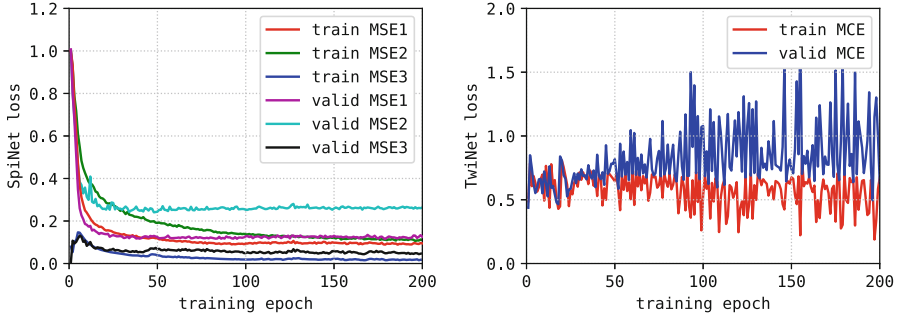


Fig. 2. Time evolution of training and validation losses.

By using TSS, we prepared a ground truth dataset named DS601 from videos recorded daily from 08:00 to 16:00 between November 2016 and April 2017. The dataset contained information about 1,014,083 vehicles. We prepared the *trainval* and *evaluation* datasets by randomly dividing DS601 in half. 80% of the *trainval* vehicles were assigned as *training* data, with the remaining 20% being *validation* data. Each time the validation loss $\mathcal{L}_{\text{TMSE}}^{\text{valid}}$ reached a new minimum, the model was saved. Finally, the evaluation processes were performed, in the same fashion as the early stopping approach [15].

Each vehicle record was described as a tuple $(\mathbf{x}, \mathbf{y}, \mathbf{t})$. The video input \mathbf{x} was a sequence of length 50, and the strain input \mathbf{y} was a sequence of length 800. \mathbf{t} is the ground truth of the vehicle speed and locus. These datasets used only the left-to-right (LtoR) subset of vehicles. Vehicles with fewer than three axles were ignored because bridge experts are interested mainly in large vehicles. Vehicles crossing the target bridge using the opposite lane were ignored, to stabilize the training process. We also ignored video data from January 6th to 15th because the sensor data during this period was lost because of a fault in the observation environment. In the end, the *trainval* and *evaluation* datasets contained 17,757 and 17,967 vehicles, respectively.

5 Experimental Results

We implemented the proposed models on Chainer¹ 4.0, and accelerated them by using CUDA² 8.0. We used the AMSGrad [16] optimizer. The batch size was 10. Figure 2 shows the time evolution of the losses over 200 epochs. Figure 3 shows the logarithmic histograms of the anomaly score. As we anticipated, a small number of responses were identified as anomalous. In this paper, we do not consider the proper thresholds for anomaly detection, but simply investigate the *anomalous* responses in detail and classify them into three classes as follows.

¹ <https://chainer.org>.

² <https://developer.nvidia.com/cuda>.

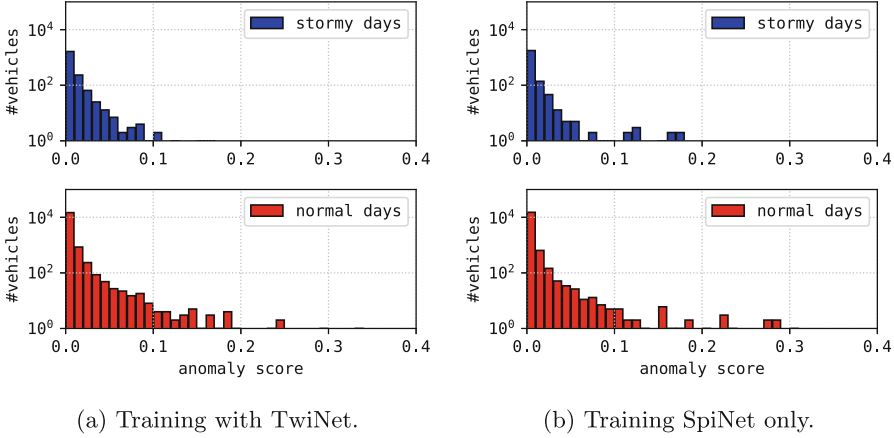


Fig. 3. Histograms for anomaly scores under bad and good weather conditions.

1. Some extremely slow vehicles may be misidentified as anomalies, in particular under snowstorm conditions. In such a situation, a vehicle takes an excessive amount of time to arrive at the sensor installation position, exceeding the four seconds allowed. SigNet will, therefore, fail to capture the strain peak caused by the vehicle. We also found cases where SigNet identified false peaks caused by other vehicles appearing just before the target vehicle arrived at the bridge entrance. To deal with such extremely slow vehicles, the input length of SigNet should be extended to enable capture of the peaks for all targets. It should be also noted that a vehicle caught in a traffic jam might not be able to drive at a steady speed, which may also affect the strain response.
2. In some vehicles, cargo may be moving about, with the resulting mechanical shock being captured by the strain sensor. This may cause ripples in the strain peak, causing the signal to resemble that of a vehicle with additional wheels. Such a situation may be observed in particular with light-loaded vehicles, but was hardly detectable from an image with a resolution of 224×224 . Because the road surface was flat, a heavily loaded cargo bed was unlikely to vibrate.
3. Some vehicles may be misidentified as anomalous due to errors in the traffic dataset. We found that some cars and motorbikes with two axles were included in the large-vehicle dataset, and were then classified as anomalies. Some cars would have more than two axles if they were towing trailers or other vehicles. Such vehicles may be identified as anomalous in this class, not because their appearance was rare, but because of their small impact on the strain response.

6 Discussion

The anomalous response detection based on spiral learning depends on features extracted from the traffic-surveillance video. Video features can be disturbed by

environmental conditions, including weather, traffic jams, pedestrians, and other vehicles in the opposite lane. For the target bridge, pedestrians were rare, and we did not observe cases where a vehicle was mistakenly detected as anomalous because of pedestrians. Vehicles in the other lane were also not a problem because the large traffic volume provided SpiNet with sufficient opportunities for learning such situations. Traffic jams may be caused by construction work, snowstorms, or traffic signals. For example, the far-side lane was closed on November 9, 2016 due to construction work. Under such conditions, the bridge behaved abnormally because many vehicles were required to reverse over the bridge. We need not be concerned with such one-off events. However, traffic signals can be a big problem in general. This problem will not be addressed in this paper, because there were no signals near our target bridge.

Figure 3(a) compares the distributions of $\mathcal{L}_{\text{MSE3}}$ for two cases, with heavy snow or haze for several hours and when conditions were fair. We examined cases that clearly seemed to involve snowy or hazy conditions, paying attention to the image sharpness. The two graphs show the robustness of the adversarial-spiral method to changing weather conditions. However, it is not clear whether this approach is necessary to achieve *weatherproof* results. The DS601 database involves both sunny and snowy days, and SpiNet might be able to obtain weatherproof results without the adversarial method. Actually, Fig. 3(b) shows such results. On the other hand, Fig. 2 shows that TwiNet failed in learning an identification function that was generalized sufficiently to detect anomalous vehicles in both the training and validation datasets. This means that SpiNet might resist TwiNet to obtain a feature space from which it is difficult to distinguish anomalous vehicles. The issue of whether adversarial learning is beneficial will be investigated in future work. Another doubt about our results was the fact that some anomalies could be detected without consulting the video data. We are confident that the video feature is necessary as an explanatory variable for strain analysis. However, there might be some anomalies whose strain response appears strange at first glance, e.g., a strain response completely buried in white noise. Failures in sensors may also cause such situations. We need to investigate in detail what type of anomaly requires the spiral learning approach. This will be revealed in future experimental work by examining those cases where SigNet guessed anomalies detected by the spiral learning approach.

7 Conclusion

We have proposed a novel anomaly detection technique for bridge deterioration analysis. The spiral learning enables a direct comparison of vehicle appearance and bridge strain responses in a common feature space. The video feature may play the role of an explanatory variable in the response. The adversarial spiral learning proposal prevented our anomaly detector from being affected by adverse weather. We tested our proposals on real observation data and identified outliers of several identifiable types. In future experimental work, we will investigate the limits on the anomaly types detectable by our method. We believe our proposals

will aid bridge damage detection by identifying anomalous strain responses whose characteristics are different from those observed during the construction period.

Acknowledgement. This work was supported by the cross-ministerial strategic innovation promotion (SIP) program (<http://www8.cao.go.jp/cstp/gaiyo/sip/>) of the Cabinet Office, Government of Japan.

References

1. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: NIPS (2014)
2. Yu, Y., Cai, C.S., Deng, L.: State-of-the-art review on bridge weigh-in-motion technology. *Adv. Struct. Eng.* **19**(9), 1511–1530 (2016)
3. Bromley, J., Guyon, I., LeCun, Y., Sackinger, E., Shah, R.: Signature verification using a “siamese” time delay neural network. In: NIPS (1993)
4. Kawakatsu, T., Kinoshita, A., Aihara, K., Takasu, A., Adachi, J.: Deep sensing approach to single-sensor bridge weighing in motion. In: EWSHM (2018)
5. Cao, M.S., Sha, G.G., Gao, Y.F., Ostachowicz, W.: Structural damage identification using damping: a compendium of uses and features. *SMS* **26**(4), 043001 (2017)
6. ZhiWei, C., QinLin, C., Ying, L., SongYe, Z.: Damage detection of long-span bridges using stress influence lines incorporated control charts. *SCTS* **57**(9), 1689–1697 (2014)
7. Huang, Y., Zhu, C., Ye, Y., Xiao, Y.: Damage detection of arch structure by using deflection influence line. In: SEEIE (2016)
8. Zaurin, R., Catbas, F.N.: Structural health monitoring using video stream, influence lines, and statistical analysis. *SHM* **10**(3), 309–332 (2011)
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
10. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: AIS-TATS (2011)
11. Caruana, R.: Multitask learning. *Mach. Learn.* **28**(1), 41–75 (1997)
12. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NIPS (2014)
13. Kawakatsu, T., Kakitani, A., Aihara, K., Takasu, A., Adachi, J.: Traffic surveillance system for bridge vibration analysis. In: IICPS (2017)
14. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: NIPS (2015)
15. Yao, Y., Rosasco, L., Caponnetto, A.: On early stopping in gradient descent learning. *Constr. Approx.* **26**(2), 289–315 (2007)
16. Reddi, S., Kale, S., Kumar, S.: On the convergence of adam and beyond. In: ICLR (2018)



CoRe: Generating a Computationally Representative Road Skeleton - Integrating AADT with Road Structure

Rohith Reddy Sankepally^(✉) and K. S. Rajan

Lab for Spatial Informatics, IIIT, Hyderabad 500032, India
rohith.reddy@research.iiit.ac.in, rajan@iiit.ac.in

Abstract. Road networks are the lifeline of a city and understanding its usage has a number of potential applications from transportation planning and engineering aspects to environmental management. While the full network is important to be analyzed for such applications, prudent planning needs one to identify the significant sections of the road network and prioritize them. Annual Average Daily Traffic (*AADT*), an estimate of the average daily traffic along a defined road segment, is one such data that helps in such endeavors. But, roads are also about connectivity and accessibility across different regions. Hence, this paper proposes a study that integrates the *AADT* data with implicit information derived from the road network to generate a computationally representative (*CoRe*) well connected sub-network, significantly smaller than the original network. While the *AADT* data analysis looks for road segments with high traffic, this paper proposes and evaluates a graph theory based approach for calculating road priorities purely based on the topological structure of the road network. The work further demonstrates the utility of the *CoRe* sub-network in terms of both, achieving gains in path computation and capturing the behavioral pattern of travelers. A case study of the *Melbourne, Australia* supports the feasibility and applicability of this knowledge integration approach.

Keywords: Knowledge discovery · Road networks · AADT
Graph theory · Skeleton · Path computation

1 Introduction

Road networks play a vital role in transportation. Roads are a popular and most preferred mode of transport influencing growth and economic activity patterns. Understanding road network behavior is crucial for transportation planning and environmental management. The primary aim of transport planning is the reduction of congestion/traffic in cities which can be achieved by diverting traffic along alternate paths. The major challenges in finding an alternate path are (a) least disturbance caused to the current network configuration (b) minimum deviation with the original path. However, construction of new roads/bridges might still

lead to their minimal usage as the activity around them may not fit into the larger dynamics of people’s travel.

A comprehensive knowledge of traffic flow is important in prioritizing roads which may help in reducing the amount of network to be analyzed for the design of alternate paths. Annual Average Daily Traffic (*AADT*) is one such data which helps in such endeavors. *AADT* is an estimate of the average daily traffic along a defined segment of roadway and is a measure used primarily in transportation planning. However as discussed, the prioritizing of roads alone does not help in efficient planning which is illustrated using the following example.

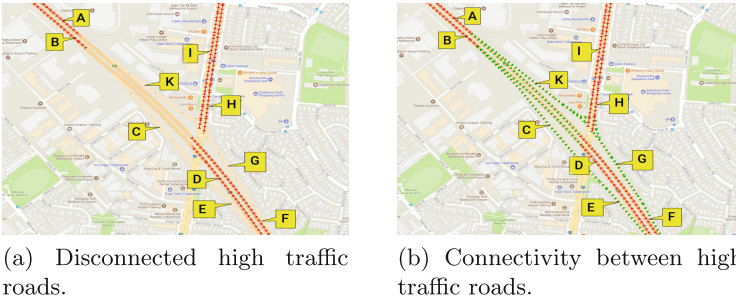


Fig. 1. Some high traffic roads in Melbourne. (Color figure online)

In Fig. 1(a), *A, B, D, F, H, I* indicate the high traffic roads in *Melbourne*. Assuming that road *H* has the highest amount of traffic, let us say one wants to manage traffic across these roads. Traditionally, one would construct a new road/bridge *J* parallel to *H* to accommodate the excess traffic on *H*. Though this may solve the problem of managing traffic partially but is not effective due to some other reasons. There may exist a situation where most of the traffic from road *H* diverges to road *F* through road *G*. In such a case, the construction of road *J* will not solve the problem because ultimately the traffic coming from either *H* or *J* will have to pass through *G* to proceed onto road *F* leading to road *G* becoming more congested. Thus it is difficult to analyze the traffic flow across these high traffic roads without a clear understanding of their connectivity.

In Fig. 1(a) we can also see that roads *C, E, G, K* are the roads connecting *A, B, D, F, H, I*. With the availability of these connections, one can predict the congestion on road *G* and thereby improve the decision of alternate path. In this case it would be better to redirect some of the traffic from road *H* through an alternate path that meets road *F* without passing through *G*. However this paper does not deal with the generation of alternate paths, but provides knowledge about connectivity between high traffic roads which can be of great use to alternate path algorithms.

In literature several algorithms were proposed in the context of knowledge extraction from road networks and traffic flow analysis. These algorithms can be broadly divided into two categories (i) Algorithms that extract knowledge using

the inherent node/edge properties of the road network (implicit knowledge). (ii) Algorithms that extract knowledge from other road network related data derived from external sources (explicit knowledge).

[1, 2] talk about various mining techniques that can be used to extract knowledge from geographical data. [3–6] use implicit network information to extract patterns like grids and other geometrical structures that can be used in the identification of important details of the road network and thereby fall into category (i). These works show that understanding the road network and its structure do help in identifying local patterns and interactions. However this understanding of road network when combined with real world data like *AADT* may provide more invaluable insights.

[7–12] use different techniques to extract patterns using data from external sources like car sharing applications, traffic sensor data to analyze traffic flow and thereby fall into category (ii). While these studies provide useful information on the traffic patterns, the interaction of these with the road network structure is, at best, marginal and does not provide a way to extrapolate those learnings. In [13], a study on the relation between the road hierarchy and traffic flow has been attempted. But, the paper does not discuss on how to identify or extract a well connected network that can help prioritize the significant sections of the network that account for majority traffic. Also, data like *AADT* while providing key insights into the denser parts of the road network, are in itself not adequate to isolate a significant, well connected part of the network, due to holes or missing connectivity across such data rich regions.

The goal of this study is to integrate the knowledge derived from *AADT* data analysis with the implicit knowledge derived from the road network to generate a computationally representative (*CoRe*) sub-network. The *CoRe Network* being significantly smaller and well connected would aid the identification of alternate paths efficiently due to the availability of high traffic roads and the connections between them. Figure 1(b) shows the *CoRe Network* containing both the high traffic roads (red color) and the roads connecting them (green color). Further, the *CoRe Network* is used in a proposed path computation model to assess its applicability in both improving path computation and capturing travel behavior.

2 Preliminaries

Graphs and Paths. A road network can be represented as a directed graph $G = (V, E)$, where V denotes a set of nodes that represent road intersections and $E \subseteq V \times V$ is the set of edges, (v_a, v_b) , each representing a road segment that connects nodes v_a and v_b . A weight function $w: E \rightarrow R$ assigns to each edge (v_a, v_b) a weight w_{ab} , which captures the cost of moving from v_a to v_b , in terms of travel time or distance. A *path* p is an ordered set of edges e_1, e_2, \dots, e_N where (e_i) is an edge $\forall i$ where $i = 1, 2, \dots, N$. A *path* between nodes v_x and v_y is denoted by $p(x \rightarrow y)$. The length $l(p)$ of a path p equals the sum of the weights of all its contained edges. $p * (x \rightarrow y)$ is a shortest path if there is no path $p(x \rightarrow y)$

such that $l(p) < l(p^*)$. The containment of an edge e in a path p defined by a function σ as follows

$$\sigma(p(x \rightarrow y), e) = \begin{cases} 1, & \text{if } e \in p(x \rightarrow y). \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

Subgraphs and Connectivity. A *connected subgraph* is a subgraph in which any two nodes are connected to each other by paths, and is connected to no additional nodes in its corresponding super graph. Any graph G contains a set of connected subgraphs $\{SG_1(V_1, E_1), \dots, SG_n(V_n, E_n)\}$.

CoRe Network. The *Skeleton Network* $G_s = (V_s, E_s)$, a *connected* subgraph of G , is an implicit knowledge derived using the inherent node/edge characteristics. The *CoRe Network* $G_{core} = (V_{core}, E_{core})$ is a refined *Skeleton Network*, combining both the characteristics of the *Skeleton Network* and traffic volume data. The *CoRe Network* is also a *connected* subgraph of G .

3 Analysis of Annual Average Daily Traffic (AADT) Data

The *AADT* data used in this study is provided by *VicRoads* [14] containing traffic volumes for freeways and arterial roads in *Victoria State, Australia* for the year 2017. The *AADT* data contained an attribute *ALLVEHS_AADT* for each road which indicated the yearly traffic volume for all vehicles. Based on the distribution of *ALLVEHS_AADT* attribute values and expert opinion, the roads are divided into 3 major classes as shown in Table 1.

Table 1. Classification of AADT Data

Class	Condition	%
Very High Traffic (VHT)	$ALLVEHS_AADT \geq 21000$	7.29
Moderate Traffic (MT)	$14000 \leq ALLVEHS_AADT < 21000$	14.68
Very Low Traffic (VLT)	$ALLVEHS_AADT < 14000$	78.01

From Table 1 we can conclude that a minority of the roads contain majority of the traffic and therefore focus should be more on analyzing these minority roads for efficient management of road networks.

4 Skeleton Network: Knowledge Based Network Extraction

4.1 Edge Priority Computation

Grid Based Division. The road network G is divided into $k \times k$ grids as shown in Fig. 2(a). The choice of k is dependent on the network density and the coverage to be achieved such that each grid has a proportional representation of nodes of interest. V^j denotes the set of nodes that belong to the j th grid.

Choice of Nodes. After the network $G(V, E)$ was divided into grids, nodes that belong to each grid are known. From each grid j a set of random nodes V_{noi}^j were chosen. The collection of chosen nodes from all the grids constitute the *nodes of interest*.

$$V_{noi} = \bigcup_{j=1}^{k^2} V_{noi}^j, \quad \text{where } V_{noi}^j \subset V^j \quad \forall j, 1 \leq j \leq k^2$$

Priority Definition. *Dijkstra* algorithm was used to compute the path between every node of interest to every other node of interest. Each edge e is assigned a value $c(e)$ which indicates the importance/priority of an edge e defined as follows

$$c(e) = \sum_{v_x \in V_{noi}} \sum_{v_y \in V_{noi}} \sigma(p^*(x \rightarrow y), e).$$

4.2 Skeleton Network Generation

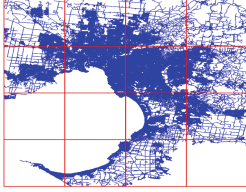
Network Selection. The *Skeleton Network* is generated based on the edge priorities $c(e)$ computed in the previous step. Based on the size of the *Skeleton Network* that is supposed to be generated a threshold value $c_{threshold}$ is chosen. The $c_{threshold}$ value helps us identify the high priority edges. Algorithm 1 is used to generate a *Skeleton Network* given a threshold value $c_{threshold}$. Figure 2(b) represents a small portion of the *Melbourne* road network with $c(e) \geq 48$.

Algorithm 1. Skeleton Network Construction Algorithm

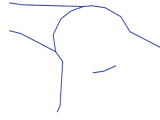
```

1: procedure CONSTRUCTSKELETON( $G, c_{threshold}$ )
2:    $G_s \leftarrow \phi$ 
3:   for  $e$  in  $G.edges()$  do
4:     if  $c(e) \geq c_{threshold}$  then  $G_s.add\_edge(e)$ 
5:    $MakeConnected(G_s)$ 
6:   return  $G_s$ 

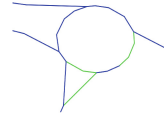
```



(a) Division of Melbourne road network into grids.



(b) A zoomed-up illustration of disconnected Skeleton Network.



(c) Connected Skeleton Network of (b) based on Algorithm 2.

Fig. 2. Extraction of Melbourne's skeleton network

Algorithm 2. Connected Network Algorithm

```

1: procedure MAKECONNECTED( $G$ )
2:    $\{SG_1(V_1, E_1), \dots, SG_n(V_n, E_n)\} \leftarrow ConnectedSubgraphs(G)$ 
3:   if  $n \neq 1$  then
4:     Let  $v_a \in SG_a, v_b \in SG_b$  such that  $a \neq b$ 
5:     for  $e$  in  $p(a \rightarrow b)$  do
6:       if  $e \notin E$  then  $G.add\_edge(e)$ 
7:     MakeConnected( $G$ )
8:   else
9:     return  $G$ 

```

Network Connectivity. According to the definition of *Skeleton Network* in Sect. 2, it is a *connected* subgraph. As illustrated in step 5 of Algorithm 1 the *Skeleton Network* is made connected. The connectivity is achieved by adding paths (with edges available in G) to G_s in order to make G_s connected as shown in Algorithm 2. Figure 2(c) represents connected *Skeleton Network* by applying Algorithm 2.

5 Evaluation of Skeleton Network with AADT Data

The *Skeleton Network* is an implicit knowledge extracted by using only the inherent node/edge characteristics of the road network. In order to assess its importance, it is evaluated using the *AADT* data.

The *AADT* data and the Open Street Map (*OSM*) data of *Melbourne* are matched using buffer analysis which is currently out of scope and not discussed in this paper. The set of edges that are matched is denoted by E_{match} . After matching, each edge e in the *OSM* road network is assigned an *AADT* attribute $aadt(e)$ which represents the level of congestion/traffic that occurs on the edge.

Given a road network $G(V, E)$, *top roads* E_{top} is defined as the roads with higher amounts of traffic and thereby higher *AADT* attribute values. A threshold $aadt_{min}$ is defined such that $E_{top} = \{e \in E \wedge aadt(e) \geq aadt_{min}\}$. The quality

of a *Skeleton Network* is defined as the fraction of the top roads E_{top} that occur in the *Skeleton Network* G_s , measured by parameter θ where $\theta = |E_s \cap E_{top}| / |E_{top}|$.

For evaluation purpose two versions of the *Skeleton Network* are generated based on their sizes as shown in Table 2. The level 1 *Skeleton Network* is evaluated against *VHT* roads while the level 2 *Skeleton Network* is evaluated against both *VHT* and *MT* roads combined. From Table 2 we observe that the majority of the roads with high traffic appear in *Skeleton Network* (both levels) indicating that the extracted implicit knowledge mimics the real world traffic volumes.

6 CoRe Network: Integration of Skeleton Network and AADT Data

From Table 2 we notice that some *VHT* roads do not appear in the *Skeleton Network*. As mentioned in Sect. 2, the *CoRe Network* is a representative connected network and comprises of roads with high traffic. Algorithm 3 describes the *CoRe Network* generation procedure. Algorithm 2 is used to make the *CoRe Network* connected. Figure 3 shows a small portion of G_{core} generated using G_s (blue color), high traffic edges (red color) and the added connections (green color).

Different versions of the *CoRe Network* are generated using different grid configurations by varying k . It was observed that the variation in k has minimal or no effect in the *CoRe Network* generated. So we proceeded by using $k = 4$ and generated the *CoRe Network* for two levels of *Skeleton Network* as given in Table 2. From Table 2 we can notice that the size of G_{core} is at most 2% more than that of its corresponding G_s which indicates that G_{core} is still significantly

Algorithm 3. CoRe Network Construction Algorithm

```

1: procedure CONSTRUCTCORE( $G_s, E_{VHT}$ )
2:    $G_{core} \leftarrow G_s$ 
3:   for  $e$  in  $E_{VHT}$  do
4:     if  $e \notin E_s$  then  $G_{core}.add\_edge(e)$ 
5:    $MakeConnected(G_{core})$ 
6:   return  $G_{core}$ 

```

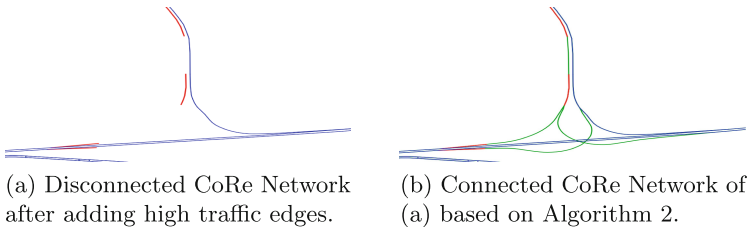


Fig. 3. Small portion of CoRe network in Melbourne (Color figure online)

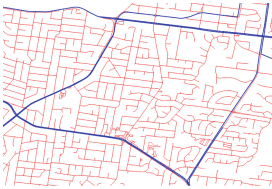
Table 2. CoRe network analysis

Level	$C_{threshold}$	$aadt_{min}$	$ E_{top} / E_{match} $	$ E_{skeleton} / E $	θ	$ E_{core} / E $
1	48	21000	0.0729	0.111	70.65	0.126
2	7	14000	0.219	0.208	80.55	0.221

smaller than G . The *CoRe Network* is generated only once as it does not change unless there are major changes to the network. Therefore the time taken for the construction of the *CoRe Network* is neglected in this study.

7 Efficient Path Computation: A Utility of CoRe Network

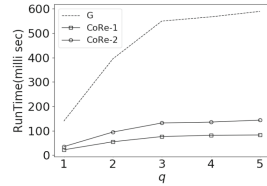
Zone Generation. From the *CoRe Network* obtained in the previous step, its corresponding *residual* network $G_{res}(V_{res}, E_{res})$ is generated where $V_{res} = V - V_{core}$ and $E_{res} = E - E_{core}$. G_{res} comprises of a set of disconnected sub-graphs $\{Z_1(V_{z1}, E_{z1}), \dots, Z_n(V_{zn}, E_{zn})\}$ called *zones*. The zones are mutually disconnected but each zone Z_j is well connected to G_{core} . Figure 4(a) shows the G_{core} (blue color) and zones (red color) formed in road network of *Melbourne*.



(a) Zones formed by the CoRe Network in Melbourne.



(b) Reduced graph used to compute path between source x and target y .



(c) Performance analysis of the proposed path computation algorithm

Fig. 4. Path computation using CoRe network (Color figure online)

Path Algorithm. Let x be the source node and y be the target node between which the path is to be computed. Let Z_x and Z_y be the *zones* to which x and y belong respectively where $x \neq y$. Given G_{core} , the path between x and y is computed on the *reduced* network $G_{xy} = Z_x \cup Z_y \cup G_{core}$. As mentioned earlier, the zones are well connected to G_{core} . Also G_{core} is a well connected subgraph as given in Sect. 2. Therefore it can be deduced that for any two nodes x and y , G_{xy} is well connected and there always exists a path between nodes x and y in G_{xy} . To compute path between any two nodes x and y , we propose a new path algorithm which performs *Dijkstra* algorithm on G_{xy} , thus achieving significant gains in path computation. Figure 4(b) shows G_{xy} with G_{core} (blue color), Z_x (green color) and Z_y (red color).

Experimental Setup. The experiments were carried out on a 64-bit *Linux* machine with an *Intel Xeon Z400* equipped with 16 GB main memory and 8 MB L3 cache. In this section we evaluate the performance of the proposed path computation algorithm. We average the path computation time over a set of 500 randomly generated source target pairs with varying path lengths. The source target pairs are generated in such a way that the distance between a node pair in the q th set lies between $\frac{(q-1) \times d_{max}}{5}$ and $\frac{q \times d_{max}}{5}$ where d_{max} is the distance between the farthest node pair in the network and $1 \leq q \leq 5$. Therefore distance between any pair of nodes in $(q+1)$ th set is greater than distance between any pair of nodes in q th set. The value of d_{max} is 142.11 kms for *Melbourne*.

Results and Observations. Figure 4(c), illustrates the variation of average path computation time with the length of the path. On Y-axis is the path computation time in milliseconds. X-axis is numbered with the query set number q where $1 \leq q \leq 5$. The curve with the dashed line represents the path computation time on G . The other two lines represent the path computation time for level 1 and level 2 of the *CoRe Network* respectively. We observe that a gain of 5–6 is achieved in path computation time while not using more than 25% of the total network. Moreover for longer distances ($q \geq 3$), the computation time is almost constant thus making the model more suitable for computation of longer paths.

In addition to improved path performance, the proposed path model also tries to capture the behavioral pattern of travelers. To understand the behavioral aspect of the path, let us consider a person traveling from source x to destination y . Usually one will first reach the main road from x , travel on the main road for a while and then divert from the main road to reach y . As stated before, G_{core} here is the set of main roads well connected to the zones. Hence, the person starting from x will first reach G_{core} with the help of roads in Z_x . He will travel on G_{core} for a while and then divert from G_{core} to reach target y with the help of roads in Z_y , thus mimicking the behavioral pattern of travelers.

8 Conclusion

In this work we presented an novel approach which integrates the implicit knowledge derived from the road network and the *AADT* data to generate a *CoRe Network*. We observe that the extracted implicit knowledge matches traffic volume data (*AADT*) to a large extent, making the model implementable in road networks even in the absence of traffic data. As *CoRe Network* is a sub network including the high traffic roads and the connectivities between them, it can be used by traffic planning applications for efficient traffic analysis. The proposed model can be used to design a time based traffic optimization by generating different versions of the *CoRe Network* on an hourly basis. A utility of *CoRe Network* is discussed in the paper by proposing a path computation model which shows gains in path computation time by exploiting the structure

of the *CoRe Network*. The path computation is faster since the network subset on which the path is computed is significantly smaller than the original network. The path computation time can further be improved by precomputing paths between every pair of nodes in G_{core} . The path computation model also captures the behavioral pattern of travelers by ensuring that the path between the source and target passes through the significant sections of the roads that are either popular or prominent connectors of different regions of a city, as the latter are part of the *CoRe Network*. Finally, we hope that this work will be useful to path planning applications and road authorities for efficient road network handling, road planning and traffic flow optimization.

References

1. Koperski, K., Han, J., Adhikary, J.: Mining knowledge in geographical data. *Commun. ACM* **26**, 65–74 (1998)
2. Koperski, K., Adhikary, J., Han, J.: Spatial data mining: progress and challenges survey paper. In: *Proceedings of the ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, Montreal, Canada, pp. 1–10, June 1996
3. Yang, B., Fan, H., Luan, X.: A multilane roads detection approach for urban transport skeleton knowledge discovery. *GeoComputation*. Wuhan University, China (2013)
4. Yang, B., Luan, X., Li, Q.: An adaptive method for identifying the spatial patterns in road networks. *Comput. Environ. Urban Syst.* **34**(1), 40–48 (2010)
5. Heinzle, F., Sester, M.: Derivation of implicit information from spatial data sets with data mining. In: *20th Congress of the International Society for Photogrammetry and Remote Sensing (ISPRS)* (2004)
6. Heinzle, F., Anders, K.H., Sester, M.: Graph based approaches for recognition of patterns and implicit information in road networks. In: *Proceedings of the 22nd International Cartographic Conference*, A Coruna, July 2005
7. Pagani, A., Bruschi, F., Rana, V.: Knowledge discovery from car sharing data for traffic flows estimation. In: *Smart City Symposium Prague (SCSP)*, pp. 1–6. IEEE, May 2017
8. Jiang, W., Vaidya, J., Balaporia, Z., Clifton, C., Banich, B.: Knowledge discovery from transportation network data. In: *Proceedings of the 21st International Conference on Data Engineering, ICDE 2005*, Chicago, pp. 1061–1072. IEEE, April 2005
9. Deng, D., Shahabi, C., Demiryurek, U., Zhu, L., Yu, R., Liu, Y.: Latent space model for road networks to predict time-varying traffic. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1525–1534. ACM, August 2016
10. Li, X., Han, J., Lee, J.-G., Gonzalez, H.: Traffic density-based discovery of hot routes in road networks. In: Papadias, D., Zhang, D., Kollios, G. (eds.) *SSTD 2007*. LNCS, vol. 4605, pp. 441–459. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73540-3_25
11. Ntoutsis, I., Mitsou, N., Marketos, G.: Traffic mining in a road-network: how does the traffic flow? *Int. J. Bus. Intell. Data Min.* **3**(1), 82–98 (2008)
12. Wang, J., Wei, D., He, K., Gong, H., Wang, P.: Encapsulating urban traffic rhythms into road networks. *Sci. Rep.* **4**, 4141 (2014)

13. Jiang, B.: Street hierarchies: a minority of streets account for a majority of traffic flow. *Int. J. Geogr. Inf. Sci.* **23**(8), 1033–1048 (2009)
14. Annual Average Daily Traffic(AADT) Data of Victoria State, Australia (2017). <https://vicroadsopendata-vicroadsmaps.opendata.arcgis.com/datasets/traffic-volume>



E-Commerce Product Recommendation Using Historical Purchases and Clickstream Data

Ying Xiao and C. I. Ezeife^(✉)

School of Computer Science, University of Windsor,
401 Sunset Ave, Windsor, ON N9B 3P4, Canada
{xiao11q, cezeife}@uwindsor.ca

Abstract. In E-commerce, user-item rating matrices for collaborative filtering recommendation systems are usually binary and sparse, showing only whether or not a user has purchased an item previously. Clickstream data containing more customer behavior have been used to improve recommendations by some existing systems referred in this paper as Kim05Rec, Kim11Rec, and Chen13Rec, using decision tree, association rule mining and category-based interest measurements respectively. However, they do not integrate valuable information from historical purchases and the consequential bond information between session-based clicks and purchases. This paper proposes Historical Purchase with Clickstream recommendation system (HPCRec), which normalizes the historical purchase frequency matrix to improve rating quality, and mines the session-based consequential bond between clicks and purchases to generate potential ratings to improve the rating quantity. Experimental results show HPCRec outperforms these existing methods, and is also capable of handling infrequent user cases, whereas other methods can not.

Keywords: E-commerce recommendation system
Collaborative filtering · CF · Clickstream history
Weighted frequent item · Data mining

1 Introduction

Recommendation systems provide suggestions of products to users, such as what items to buy, what music to listen to, or what online news to read [12]. Collaborative filtering (CF) is the most popular method used for recommendations, where a user's preferences can be associated with the preferences of a similar user community. In E-commerce, CF usually takes a binary user-item rating matrix (e.g., Table 1) as input, where “1” indicates that a user has purchased a

C. I. Ezeife—This research was supported by the Natural Science and Engineering Research Council (NSERC) of Canada under an Operating grant (OGP-0194134) and a University of Windsor grant.

product before, and “?” means a user has not. Given an incomplete user-rating matrix R of m users for n items with missing ratings (r_{uj}) of item j for user u , the user-based neighborhood model CF [1] would predict the unknown ratings (r_{uj}) through the following steps: (1) computing the mean rating for each user u ; (2) calculating the similarity between target user u and all other users v ; (3) computing user u 's peer group P ; and then (4) predicting rating for target user u for item j .

The user-item rating matrix (eg., Table 1) in E-commerce usually contains part information of the historical transaction records (e.g., Table 2). In Table 2, each row records a purchase (a collection of item ids) that happened in a session and there may be multiple products for each purchase. There are some known fundamental issues with this approach as follows: (1) cold start: when new items or new users with unknown ratings/preferences appear in the database; (2) sparsity issue: When the known rating data takes only a very small proportion in the user-item rating matrix, (only a few hundreds of billions of products purchased by users) leading to confusing and compromised recommendations; (3) scalability issue: As the numbers of users and products grow rapidly, the time complexity and space complexity issues become more prominent.

Methods for enhancing recommendation input data have been studied in various papers, such as [4, 8, 9], since the user-item rating matrix in e-commerce only shows what items a user has purchased previously, which does not provide a lot of information about customer purchase history or item purchase history for the purposes of improving recommendation accuracy. In addition to the rating matrix, some other data sources such as clickstream data, meta data and transactions have been discovered and utilized to improve recommendations. Clickstream data have been used to predict a user's next request [5], discover patterns to build profile for customers [11], find the possibilities of purchasing items [14]. **Transaction data** (Table 2) is the detailed purchase history where each row records item purchases that occurred in a session by a specific user. **User-item rating matrix** (Table 1) is the basic input data format for CF recommendation systems. **Clickstream data** is the electronic record of a user's activity on the Internet [3]. In e-commerce, clickstream data reflects a user's Internet foot-

Table 1. A user-item rating matrix

Customer\Item	1	2	3	4
1	?	1	1	?
2		1	1	?
3		1	?	?

Table 2. A historical transaction table

SessionId	UserId	Purchases
1	1	2
2	1	2, 3
3	2	1, 2, 4
4	2	2, 4, 4
5	3	1
6	3	

prints for behaviors such as clicks, basket placement, purchases, reading reviews and so on. Clickstream events may include attributes like SessionId, UserId, ItemId, Category, Time, visit duration, visit types and IP address. **Meta data** is the description of user preferences (such as product categories purchased (eg. automotive, beauty, electronics, software); and product details (eg., OS is iOS, item weight is 136 g, color is Black, etc.)). These information can be used in the recommendation engines for comparing products and recommending similar products.

1.1 Observations and Assumptions

Two assumptions are given in this section based on some observations.

Data Sparsity. CF method in E-commerce suffers from data sparsity of the rating matrix given the large number of products. It only uses the binary user-item rating purchase matrix (Table 1) which does not reflect much regarding: (1) how much a user likes an item; (2) how frequently or how long ago a user purchased an item; (3) what quantity of a product was purchased. This information is not integrated in the CF user-rating matrix but can potentially improve recommendations accuracy.

Information Distribution. Traditional CF systems only take purchase data into calculation, despite the fact that there are other data available for analysis. Moreover, from the data provided by [ACM RecSys 2015 \[2\]](#) in the flies yoochoose-clicks and yoochoose-buys. We can observe that the click data (yoochoose-clicks) is almost 27 times more than the purchase data (yoochoose-buys).

Consequential Relationship between Clicks and Purchases. Clickstream data and the purchase data (Table 2) can be re-organized to include these user click sequences as well as products purchased during each session as in Table 3. Sometimes there are no purchases but only clicks in a session. The relationship between clicks and purchases is called consequential relationship because clicks lead to certain purchases.

Assumption 1. Need to improve the rating quality. The binary user-item rating matrix in Table 1 is less informative compared to the original transaction records in Table 2 as it does not indicate how frequently an item is purchased. If there is a way to extract more information from the transaction records into the rating matrix to capture missing data, the recommendation system would perform better with the more informative input data. We first form a user-item purchase frequency matrix (Table 4) from Table 2, where each value represents the amount of a product purchased by a user. We then normalize the purchase frequency to a scaled value (0 to 1 in Table 5) representing how interested a user is in one item as compared to various others, the formula is introduced in Sect. 3.1.

Assumption 2. Need to improve the rating quantity. In the sessions with purchases where purchases were made by a user in Table 3, the interest of the user

Table 3. Consequential table

SessionId	UserId	Clicks	Purchases
1	1	1, 2	2
2	1	3, 5, 2, 3	2, 3
3	2	2, 1, 4	1, 2, 4
4	2	4, 4, 1, 2	2, 4, 4
5	3	1, 2, 1	1
6	3	3, 5, 2	

Table 4. User-item purchase frequency matrix

Customer\Item	1	2	3	4
1	?	2	1	?
2	1	2	?	3
3	1	?	?	?

for the purchased products is affirmative, whereas for the sessions without a purchase, we cannot determine whether a user is interested in a product. However, the clicks in a session without a purchase may imply potential interest. We mainly discover the session-based consequential bond to generate more potential rating scores. For example, for session 6 where user 3 clicked products 3, 5 and 2 in Table 3, if we can find sessions sharing a similar click pattern but has purchased some product, then we can use the possibility to enrich the rating matrix such as Table 6 which is less sparse than the original table (Table 1). We assume by integrating the consequential bond information to the user-item purchase matrix, the accuracy of recommendations will be improved.

Table 5. Normalized user-item purchase frequency matrix

Customer\Item	1	2	3	4
1	?	0.89	0.45	?
2	0.27	0.53	?	0.8
3	1	?	?	?

Table 6. Enriched and normalized user-item purchase matrix

Customer\Item	1	2	3	4
1	?	0.89	0.45	?
2	0.27	0.53	?	0.8
3	1	1	0.189	0.167

1.2 Paper Contributions

Studies in [3,10,13] have implied that there is a consequential relationship between behaviors collected as clickstream data and purchase data. This paper proposes the HPCRec system, which enriches the rating matrix from both quantity (converting some initial ratings of 0 to a consequential bond values) and quality (converting some initial ratings of 1 to a value between 0 and 1 that includes frequency of purchase of an item) aspects. The enriched matrix is then processed using the CF method. It takes the consequential table (eg., Table 3) and user-item purchase frequency matrix (eg., Table 4) as input, follows four main steps below to output a rating matrix with predicted ratings.

1. Normalizing user-item purchase frequency matrix to a new user-item rating matrix using unit vector formula with details in Sect. 3.1.
2. In the consequential table, for each session without a purchase belonging to a user, find the top-N similar sessions with purchases by comparing the click sequences using function CSSM (Clickstream Sequence Similarity Measurement) in Sect. 3.2. Then use the similarity as weight and assign to the purchases in the selected top-N session. This step generates a weighted transaction table where weights are similarities assigned to purchases.
3. Using the weighted transaction table from step 2, call function TWFI (Transaction- based Weighted Frequent Item) in Sect. 3.3 to get a list of items with purchasing possibilities.
4. For each item from the previous step, if the user from step 2 has not previously purchased the product, enrich the resulting matrix from step 1 with the possibility. Then, return to step 2 for the next session without a purchase if possible, and otherwise continue to step 5.
5. With the enriched rating matrix, run the CF algorithm and predict ratings. Return the rating matrix with predicted ratings.

The new feature contributions of paper include:

- (i) **Improving the quality of ratings** by capturing the level of interest in a product already purchased by a user before through record of normalized frequency of purchase using the method in [15].
- (ii) **Improving the quantity of ratings** with consequential bond between clicks and purchases, for the sessions without purchases.
- (iii) **Improving the recommendation accuracy** by processing the enriched rating matrix generated in the CF algorithm.
- (iv) **Making recommendations for infrequent users.** Our HPCRec system performs a session-based interest mining algorithm which finds the interests of the most similar sessions compared to the existing sessions without a purchase for the user.

1.3 Paper Outline

Section 2 shows some important research related to integrating clickstream data to make better recommendations; Sect. 3 proposes HPCRec system. Section 4 discusses experiments, and Sect. 5 presents conclusions and future work.

2 Related Work

Some important work have been done to integrate clickstream data into recommendation systems. A few popular related strategies is discussed in this section. noindent **A stage-based decision tree approach** [9]. Kim05Rec [9] first forms a decision tree for basket placement based on behaviors such as searching, browsing and click times. For each path in the tree, it gives the proportion of users taking that path and uses it as the probability to enhance the user-item matrix

with these basket placement probability data before running CF algorithm to predict ratings. Kim05Rec [9] improves accuracy, but it shrinks the candidate range by only taking the products which have been previously paid attention to, only focuses on major cases by always choosing the popular path in the decision tree. **An association rule approach** [8]. Kim11Rec [8] integrated with association rule mining calculates the confidence between products in different stages including click, basket placement, and purchase. For instance, the clicks ($\langle abc \rangle$, $\langle bcd \rangle$, $\langle efa \rangle$) that happened in the click stage for three different sessions, and there are some items such as a and b that a user has shown interest during current session, then the system finds the most relevant clicked item in (c, d, e, f) for a and b by comparing the lift score. Same for the basket placement and purchase stages. The next step is assigning weights to the scores of three different stages to calculate a final score. It was proven to outperform the decision tree approach, but by applying association rule mining, it loses the connection between users sharing special interests. **A category-based common interest approach** [4] Chen13Rec finds the similarity between two users on their clickstream sequences to address a better neighborhood with similar interest. It first uses the longest common subsequence to compare two click sequence groups of two users; the second indicator is the similarity between user-product click frequency vectors showing the click times of a user for all products; the third indicator is the similarity between user-product visiting duration vectors. By selecting top-N similar users using three indicators, the CF method can use it for neighbor selection and improve the poor relationship between users in the rating matrix.

3 Proposed HPCRec System

This paper proposes a novel recommendation system HPCRec (Algorithm 1) which integrates purchase frequencies and the consequential bond relationship between clicks and purchases. By processing this information, it enhances the user-item rating matrix in both quantity and quality aspects and then improves recommendations. We use pre-processed consequential table (Table 3) showing user click sequences with their purchases and frequency matrix (Table 4) showing number of times an item is purchased as input, and HPCRec returns a matrix with predicted ratings (Table 7). There are three functions FN (Frequency Normalization), CSSM and TWFI used in HPCRec. HPCRec was also proven to give better recommendations to infrequent users. The HPCRec (Algorithm 1) is explained with example Tables 3 and 4 as input:

1. Normalize the purchase frequency in Table 4 for each user on each item, and get a normalized rating matrix by applying unit vector formula (Eq. 1) of Sect. 3.1 as Table 5;
2. For each session without a purchase, such as session 6 for user 3 in Table 3. Calculate the similarity between session 6 and other sessions with purchases (1,2,3,4,5) by comparing the clicks calling CSSM in

Algorithm 1. HPCRec System to Predict Ratings

Input: C , consequential table; F , frequency matrix
Output: P , a rating matrix with predicted ratings

- 1: M , normalized rating matrix \leftarrow FN(F) in Section 3.1;
- 2: **for all** N , session without a purchase \in consequential table **do**
- 3: T , weighted transaction table \leftarrow null;
- 4: **for all** Y , session with purchase \in consequential table **do**
- 5: similarity \leftarrow CSSM($N.clicks$, $Y.clicks$) in Section 3.2;
- 6: add (similarity, $Y.purchases$) to T ;
- 7: **end for**
- 8: Is , weighted frequent items \leftarrow TWFI(T) in Section 3.3;
- 9: **for all** I , weighted frequent item $\in Is$ **do**
- 10: **if** M does not contain ratings for ($N.user, I.item$) **then**
- 11: add ($N.user, I.item, I.weight$) to M ;
- 12: **end if**
- 13: **end for**
- 14: **end for**
- 15: P , rating matrix with predicted ratings \leftarrow CF(M);
- 16: **return** P ;

Sect. 3.2, get $CSSM(\langle 3, 5, 2 \rangle, \langle 1, 2 \rangle) = 0.37$, $CSSM(\langle 3, 5, 2 \rangle, \langle 3, 5, 2, 3 \rangle) = 0.845$, $CSSM(\langle 3, 5, 2 \rangle, \langle 2, 1, 4 \rangle) = 0.33$, $CSSM(\langle 3, 5, 2 \rangle, \langle 4, 4, 1, 2 \rangle) = 0.245$, $CSSM(\langle 3, 5, 2 \rangle, \langle 1, 2, 1 \rangle) = 0.295$; form a weighted transaction table using the similarity as purchases and weight as transaction records such as $[(\langle 2 \rangle : 0.37), (\langle 2, 3 \rangle : 0.845), (\langle 1, 2, 4 \rangle : 0.33), (\langle 2, 4, 4 \rangle : 0.245), (\langle 1 \rangle : 0.295)]$;

3. Call TWFI in Sect. 3.3 with the weighted transaction table from step 2, and get weighted frequent items (2:1, 3:0.189, 4:0.167); for all weighted frequent items, if the user has not purchased it, add the possibility into the normalized frequency matrix such as in Table 6.
4. Return to step 2 if there is more session without a purchase, otherwise, run the CF algorithm using the updated rating matrix (Table 6) to get predicted ratings for all of the original unknowns as demonstrated in Table 7, return the rating table with predicted ratings. Accuracy also can be calculated.

Table 7. User-item rating matrix with predicted ratings

Customer\Item	1	2	3	4
1	0.63	0.89	0.45	0.49
2	0.27	0.53	0.35	0.8
3	1	0.74	0.27	0.33

3.1 FN: Frequency Normalization

In this module, we take the user-item purchase frequency (Table 4) as input, normalize the frequencies into numbers between 0 and 1 using the unit vector formula [15] (Eq. 1). For each user, $\langle x_1, x_2, x_3, \dots, x_n \rangle$ is the purchase vector showing the purchase frequency of product 1, 2, 3, \dots , n respectively. For user 2, the purchase vector is $\langle 1, 2, 0, 3 \rangle$, so the normalized purchase frequency for user 2 on item 2 is $\frac{2}{\sqrt{1^2+2^2+0^2+3^2}} = 0.53$. The normalized frequency matrix is in Table 5, from which we can see that for each user, the differences between ratings reflects the different levels of interest. We also tried the feature scaling normalization method (Eq. 3) to normalize the frequencies, but the unit vector formula was more effective.

$$x' = \frac{x}{\sqrt{x_1^2 + x_2^2 + x_3^2 + \dots + x_n^2}}. \quad (1)$$

3.2 CSSM: Clickstream Sequence Similarity Measurement

Inspired by the idea of Chen in [4], we introduce CSSM (Clickstream sequence similarity measurement) which takes the frequency and position of items in sequences into consideration to calculate the similarity. Instead of calculating the category visiting sequences and frequencies, CSSM calculates product click sequences and frequencies. We explain this function in steps using two click sequences $\langle 3, 5, 2 \rangle$ and $\langle 3, 5, 2, 3 \rangle$ in Table 3 as an example.

1. Calculate the longest common subsequence rate $LCSR(x, y) = \frac{LCS(x, y)}{\max(|x|, |y|)}$, where the longest common subsequence (LCS) [7] is defined in Eq. 2. e.g., $LCS(\langle 3, 5, 2 \rangle, \langle 3, 5, 2, 3 \rangle) = 3$, the maximum sequence size is 4, so $LCSR(\langle 3, 5, 2 \rangle, \langle 3, 5, 2, 3 \rangle) = 3/4$;

$$LCS(X_i, Y_j) = \begin{cases} \emptyset & \text{if } i = 0 \text{ or } j = 0 \\ LCS(X_{i-1}, Y_{j-1}) \cup x_i & \text{if } x_i = y_j \\ \text{longest}(LCS(X_i, Y_{j-1}), LCS(X_{i-1}, Y_j)) & \text{if } x_i \neq y_j \end{cases} \quad (2)$$

2. Calculate the item frequency similarity (FS). First, form a distinct itemset containing all the items in both sequences, eg., $\langle 2, 3, 5 \rangle$ in this example. For each sequence, form a vector of frequency for the items in itemset, $\langle 1, 1, 1 \rangle$ for $\langle 3, 5, 2 \rangle$, and $\langle 1, 2, 1 \rangle$ for $\langle 3, 5, 2, 3 \rangle$; then find the cosine similarity between two vectors, which is 0.94 in this case;
3. Compute the final similarity $Sim = \alpha \times LCSR + \beta \times FS$, where $\alpha + \beta = 1, 0 < \alpha, \beta < 1$, α and β are weight to balance the two indicators from step 1 and 2. In the real procedure, we train our dataset with different α and β to find the best combination for prediction. If set $\alpha = 0.5, \beta = 0.5$, the final similarity $Sim(\langle 3, 5, 2 \rangle, \langle 3, 5, 2, 3 \rangle) = 0.5 \times 3/4 + 0.5 \times 0.94 = 0.845$ in the example.

3.3 TWFI: Transaction-Based Weighted Frequent Item

This function takes a weighted transaction table where weights are assigned to each transaction as input, and returns items with weighted support in a given threshold. We explain this with an example [$\langle(2) : 0.37\rangle, \langle(2, 3) : 0.845\rangle, \langle(1, 2, 4) : 0.33\rangle, \langle(2, 4, 4) : 0.245\rangle, \langle(1) : 0.295\rangle$], $\text{MinWeightedSupport} = 0.15$, where each unit has pattern and weight in such form $\langle(\text{item ids in a transaction}) : \text{weight}\rangle$.

1. Calculate support. Form a distinct item set from all the transactions, and find the support for each item, e.g., $\langle 1 : 2, 2 : 4, 3 : 1, 4 : 3\rangle$;
2. Compute the average weighted support ($\text{AWS} = \text{AW} \times \text{support}$) for each item using the same strategy in [16], where average weight ($\text{AW} = \frac{\text{sum}(\text{weight})}{\text{support}}$), which makes $\text{AWS} = \text{sum}(\text{weight})$, e.g., $\text{AWS}(4) = 0.33 + 0.245 + 0.245 = 0.82$, $\langle 1 : 0.625, 2 : 1.79, 3 : 0.845, 4 : 0.82\rangle$; We also tried using maximum weighted support ($\text{MWS} = \max(\text{weight}) \times \text{support}$), $\text{AWS}(4) = \max(0.33, 0.245, 0.245) = 0.33 \times 3 = 0.99$, the maximum approach was proven good, but the average approach is better.
3. Normalize weighted support using feature scaling (Eq. 3), so for the average weighted support, $\max = 1.79$, $\min = 0.625$, then the new average weighted support for item 3 is $\frac{(0.845 - 0.625)}{(1.79 - 0.625)} = 0.189$, all the weighted supports are $\langle 1 : 0, 2 : 1, 3 : 0.189, 4 : 0.167\rangle$;

$$x' = \frac{x - \min}{\max - \min}. \quad (3)$$

4. Return all the items with normalized weighted support greater or equal to $\text{MinWeightedSupport}$, e.g., $\langle 2:1, 3:0.189, 4:0.167\rangle$ for using average weighted support;

4 Evaluation and Comparative Analysis

We have implemented HPCRec and compared with some existing approaches, and it has been proven that HPCRec is capable of making recommendations for infrequent users which the existing systems can not. Experimental results show that HPCRec is more accurate which proves that the consequential bond with the normalized frequencies are more effective at predicting user interest.

4.1 An Example for Handling Infrequent User Cases

Here we use a small example (Tables 9 and 8) to explain that HPCRec can handle the infrequent case. There are 100 sessions in this example, user 100 intends to purchase “g” after reviewing “g” repeatedly for three times, but only HPCRec can predict that. The weak bond between infrequent users is ignored by other approaches. We find the most similar clicks sequence for “ggg” which is “fgh”, the similarity between them maybe weak, but it is the strongest for “ggg”, then we use the purchases in session 99 as recommendations, which successfully finds item “g” for user 100.

Table 8. Product table

ItemId	1	2	3	4	5	6	7	8
ItemName	a	b	c	d	e	f	g	h
Category	1	2	2	1	3	4	1	4

Table 9. Consequential table

SessionId	UserId	Clicks	Purchases
1	1	abcdade	ace
2	2	cdea	acdd
3	3	ddcabe	aed
4-98	4-98	abce	ace
99	99	fgh	fg
100	100	ggg	

4.2 Experimental Design

To make sure the evaluation is fair, we only select the top-N (N is productNumber/10) scores from different approaches and normalize the scores using Eq. 3 as the rating to keep the measuring standard consistent. Then we feed the new rating matrix to an evaluation method of an existing recommendation library Librec [6] to test all of the approaches. For Chen13Rec [4], we use the measured relationship to enhance the similarity table during the evaluation. For HPCRec, we ran through using both average weighted and maximum weighted support strategies in module TWFI (Sect. 3.3).

Dataset and Sample Selection: We use the dataset provided by YOO-CHOOSE GmbH for ACM RecSys 2015 [2], which is from an online retailer in Europe. There are two files recording 33,040,175 clicks and 1,177,769 purchase events respectively, all of the events happened in 9,512,786 unique sessions, the total amount of product is 52,739 belonging to 339 categories. For sample selection, we randomly select a certain amount of session 10 times and use the average value. For each time, given the lack of user information in the dataset, we generate a reasonable number of user and assign to the sessions randomly then use the average value from 10 times attempts. It has been proven that regardless of how users are distributed in sessions, our methods are better.

Method: We evaluate with both user-based and item-based CF recommendations. Itemknn selections and pcc similarity method are used for item-based evaluation, userknn and cosine similarity method are used for user-based evaluation. We use evaluation measurements AP (Average Precision), Precision and Recall from Librec [6]. For the calculation details, please visit Librec.

4.3 Experimental Results

From both user-based (Fig. 1) and item-based (Fig. 2) CF evaluation results on varying numbers of sessions, we can see the accuracy keeps dropping as the amount of sessions increases, our approaches are still better in this respect. For average accuracy and recall, our methods significantly beats others. We select a different number of top-N scores from all of the methods for calculation and evaluation. Both user-based CF (Fig. 3) and item-based CF (Fig. 4) are still the best which proves the high quality of our scores. Kim05Rec [9] also demonstrates good performance.

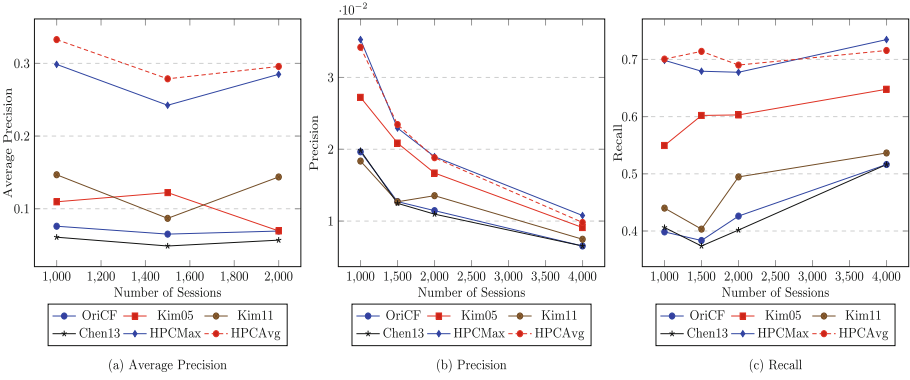


Fig. 1. User-based CF Evaluation on different number of sessions

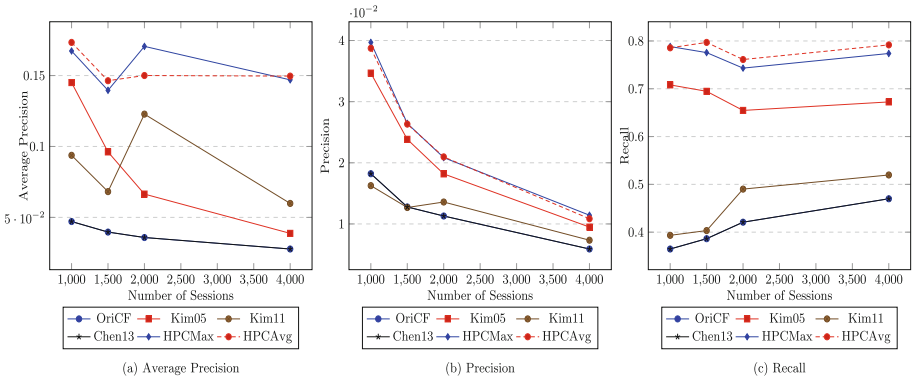


Fig. 2. Item-based CF Evaluation on different number of sessions

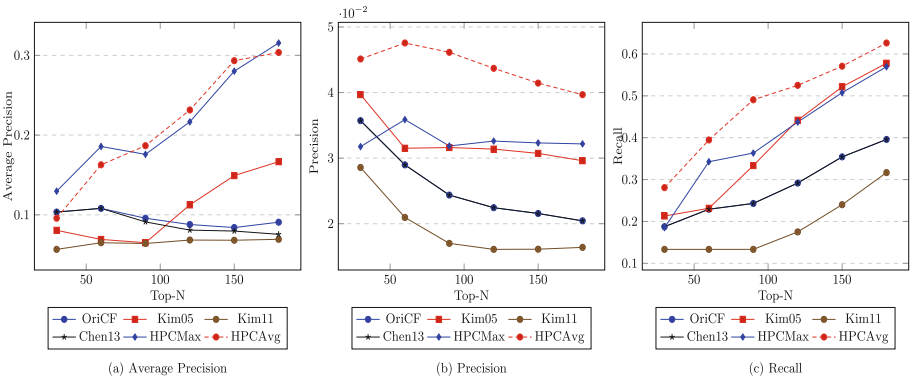


Fig. 3. User-based CF Evaluation on different number of top-N scores

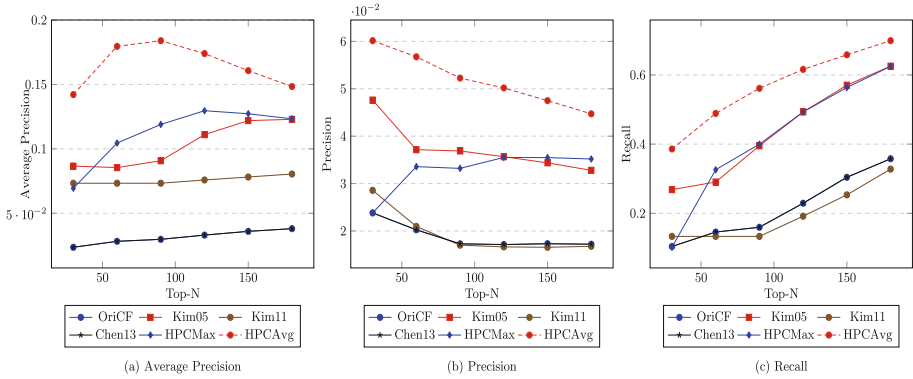


Fig. 4. Item-based CF Evaluation on different number of top-N scores

5 Conclusions and Future Work

To conclude, proposed system HPCRec enhances the quantity and quality of ratings of the user-item matrix by integrating historical purchases and click-stream data, therefore improves the recommendation qualities. Our experimental results show that our approaches outperform some existing systems referred in this paper. For future work, we plan to mine more information out of the historical data to improve recommendations such as how long ago a user purchased an item, and the frequent sequential purchase patterns, toward incorporating multiple data sources.

References

1. Aggarwal, C.C.: Recommender Systems. Springer, Cham (2016). <https://doi.org/10.1007/978-3-319-29659-3>
2. Ben-Shimon, D., Tsikinovsky, A., Friedmann, M., Shapira, B., Rokach, L., Hoerle, J.: Recsys challenge 2015 and the yoochoose dataset. In: Proceedings of the 9th ACM Conference on Recommender Systems, pp. 357–358. ACM (2015)
3. Bucklin, R.E., Sismeiro, C.: Click here for internet insight: advances in clickstream data analysis in marketing. *J. Interact. Mark.* **23**(1), 35–48 (2009)
4. Chen, L., Su, Q.: Discovering user’s interest at e-commerce site using clickstream data. In: 2013 10th International Conference on Service Systems and Service Management (ICSSSM), pp. 124–129. IEEE (2013)
5. Gündüz, Ş., Özsu, M.T.: A web page prediction model based on click-stream tree representation of user behavior. In: Proceedings of the ninth ACM SIGKDD, pp. 535–540. ACM (2003)
6. Guo, G., Zhang, J., Sun, Z., Yorke-Smith, N.: LibRec: a java library for recommender systems. In: UMAP Workshops, vol. 4 (2015)
7. Hunt, J.W., MacIlroy, M.D.: An Algorithm for Differential File Comparison. Bell Laboratories, Murray Hill (1976)
8. Kim, Y.S., Yum, B.J.: Recommender system based on click stream data using association rule mining. *Expert Syst. Appl.* **38**(10), 13320–13327 (2011)

9. Kim, Y.S., Yum, B.J., Song, J., Kim, S.M.: Development of a recommender system based on navigational and behavioral patterns of customers in e-commerce sites. *Expert Syst. Appl.* **28**(2), 381–393 (2005)
10. Moe, W.W., Fader, P.S.: Dynamic conversion behavior at e-commerce sites. *Manag. Sci.* **50**(3), 326–335 (2004)
11. Park, Y.J., Chang, K.N.: Individual and group behavior-based customer profile model for personalized product recommendation. *Expert Syst. Appl.* **36**(2), 1932–1939 (2009)
12. Ricci, F., Rokach, L., Shapira, B.: Introduction to recommender systems handbook. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) *Recommender Systems Handbook*, pp. 1–35. Springer, Boston (2011). https://doi.org/10.1007/978-0-387-85820-3_1
13. Sismeiro, C., Bucklin, R.E.: Modeling purchase behavior at an e-commerce web site: a task-completion approach. *J. Mark. Res.* **41**(3), 306–323 (2004)
14. Van den Poel, D., Buckinx, W.: Predicting online-purchasing behaviour. *Eur. J. Oper. Res.* **166**(2), 557–575 (2005)
15. Weisstein, E.W.: Normal Vector: From MathWorld—A Wolfram Web Resource. <http://mathworld.wolfram.com/NormalVector.html> (2002)
16. Yun, U., Leggett, J.J.: WFIM: weighted frequent itemset mining with a weight range and a minimum weight. In: *Proceedings of the 2005 SIAM International Conference on Data Mining*, pp. 636–640. SIAM (2005)



Effective Classification of Ground Transportation Modes for Urban Data Mining in Smart Cities

Carson K. Leung^(✉) , Peter Braun, and Adam G. M. Pazdor

University of Manitoba, Winnipeg, MB, Canada
kleung@cs.umanitoba.ca

Abstract. The increasing amount of digital data in urban research has drawn attention in *urban data mining*. In urban research (e.g., travel studies in urban areas), researchers who conduct paper-based or telephone-based travel surveys often collect biased and inaccurate data about movements of their participants. Although the use of global positioning system (GPS) trackers in travel studies improves the accuracy of exact participant trip tracking, the challenge of labelling trip purpose and transportation mode still persists. The automation of such a task would be beneficial to travel studies and other applications that rely on contextual knowledge (e.g., current travel mode of a person). In this paper, we focus on transportation mode classification. In particular, we develop a system that improves classification accuracy of ground transportation modes (e.g., bus, car, bike, or walk). When compared with related works, our system increases the classification accuracy by uniquely using GPS and accelerometer data together with a window history queue (which uses previously encountered data). Evaluation results show that our system achieves a high classification accuracy.

Keywords: Data mining · Classification · Ground transportation mode
Global positioning system (GPS) data
Geographic information system (GIS) data · Accelerometer data
Window history queue (WHQ)

1 Introduction

In the current era of big data, huge volumes of a wide variety of data of different veracity (e.g., uncertain and imprecise data [1–3]) can be easily collected or generated at a high velocity in many real-life applications. Embedded in these big data are valuable information or knowledge. This calls for *data mining* [4–6], which aims to extract implicit, previously unknown, and potentially useful information from a large amount of data [7]. With the increasing amount of digital data in urban research, the field of *urban data mining* [8–12]—which aims to discover knowledge from data related to urban problems for solving urban issues—has also developed and been given more attention.

In urban research (specifically, travel studies in urban areas), researchers often have been using paper-based and telephone-based *travel surveys* [13]. Those travel surveys

can often be biased and contain inaccurate data about movements of their participants. Participants tend to under-report short trips and irregular trips. Additionally, car trips are often reported to be shorter than they are, and public transit trips are reported to be longer than they are [14, 15].

Commute diaries [16, 17] are another approach of collecting data about people's daily commutes, but they also have been shown to be prone to errors. When people are asked to use a diary to keep track of their commutes, they often forget to record their commutes throughout the day. When trips are recorded at the end of the day, diary studies can then inherit the same problems as paper-based and telephone-based travel surveys. Moreover, commute diary studies can also be a mental burden to study participants and cannot be used long term [18]. As people's willingness to record trips accurately throughout the day declines with each day of participation, the accuracy of the commute diaries also drops accordingly [19].

To avoid data collection problems associated with the paper-based travel survey and commute diaries, there is now a shift towards the use of global positioning system (GPS) trackers to collect more objective commute data from participants. Studies show that *GPS-based travel surveys* [20, 21] are more accurate than the aforementioned surveys and diaries.

Although the use of GPS trackers in travel studies improve the accuracy of exact participant trip tracking, the challenge of *labelling trip purpose* and *classifying transportation mode* still persists. Nowadays, with the use of electronic trackers, researchers are often dealing with a large amount of movement trajectories that are collected by participants of a study who use GPS trackers or other sensors (e.g., Bluetooth, Wi-Fi, accelerometers, barometers, etc.). Manual segmentation of trajectories based on transportation mode is a labor-intensive task, and most likely infeasible when performed on big data [22]. The automation of such a task would obviously be beneficial to travel studies and other applications that rely on contextual knowledge (e.g., current travel mode of a person). As an example for a contextual use for transportation, when the person is driving, a device like a smartphone could recognize that the person is driving in a car and give a notification about the current estimated time of arrival—assuming that the phone knows the destination based on previous user interaction or saved frequently visited locations. Another application for transportation mode classification is the automatic trip transportation mode labeling for trip history. This is similar to timeline in Google Maps (which keeps track of a user's location history and attempts to automatically classify trips with the major transportation mode). However, the resulting classifications are observed to be not very accurate and need a lot of corrections by the user. Moreover, it does not track when transportation modes were changed. Hence, a more accurate algorithm or system is needed.

By using a standalone tracking and logging device, participants of travel surveys would be able to log sensor data reliably and consistently because developers and engineers have full control over the device and the hardware. Moreover, software platforms are the same on every device. Such loggers can be used to log data to local device storage, and then collect the logged data for data retrieval. Some devices could also connect to a smartphone application on a participant's phone via Bluetooth and collect data on regular intervals. The collected data could be further processed, and the users could be prompted with surveys. In such a case, transportation mode

classification could happen on a smartphone. By doing so, computational burden on the logger device is reduced, a cheaper architecture that requires weaker processing units is possible, power consumption is potentially decreased, and thus the battery life is increased.

As a preview, the system proposed in this paper works well for all scenarios described above—*smartphone logging (with online and offline classification)* and *standalone logging devices*. Moreover, in this paper, our system focuses on the following:

- offline learning (with which the classification model is usually trained on the server);
- offline classification/prediction (with which the classification model classifies the ground transportation modes on the server), and
- online classification/prediction (with which the classification model classifies the ground transportation modes on the mobile device).

In order to increase classification accuracy, our transportation mode classification system not only uses the basis of an online classification approach, but also uses multiple windows of previously encountered data. Online classification focuses on one window at a time during processing and classification. To the best of our knowledge, existing academic works in this area have yet not taken advantage of previously encountered data windows in order to increase the classification accuracy of the currently processed window of data. Hence, a natural question to ask is: Is it possible to compute features based on previously seen data for a single trip and use it to significantly improve real-time window based ground transportation mode classification?

In this paper, we design new classification features based on a *window history queue (WHQ)*, which focuses on summarizing data from previously encountered data windows. Our goal is to improve accuracy of transportation mode classification when compared with existing systems. Our **key contribution** is our classification system for ground transportation modes. To the best of our knowledge, uniquely using new features based on a WHQ, together with accelerometer data and GPS data, in a single system during the classification process improves the classification accuracy.

The remainder of this paper is organized as follows. The next section discusses related works. Section 3 presents our proposed transportation mode classification system. Evaluation results and conclusions are given in Sects. 4 and 5, respectively.

2 Related Works

Most research works [23–25] related to transportation mode classification require at a minimum GPS or accelerometer data for creating a classification model. Much research exists [26], where researchers used some combination of data from different sensors (e.g., GPS, accelerometers) and other modern smartphone sensors (e.g., barometer, magnetometer, etc.). Some researchers [27] also augmented geographic information system (GIS) data to their sensor datasets. However, none of them uniquely combined GPS, accelerometer and GIS data in a single system. In contrast, our system combines GPS, accelerometer and GIS data.

To elaborate, Zheng et al. [23] used supervised decision trees and graph based post-processing after classification to classify transportation modes from *GPS data only*.

In contrast, Hemminki et al. [24] focused on classifying transportation modes (“stationary”, “walk”, “bus”, “train”, “metro”, “tram”) with the use of *only accelerometer data*. To classify transportation modes, three different classifiers were trained with a combination of AdaBoost and Hidden Markov Model for three different classes of modes. Shafique and Hato [25] also used accelerometer data only. They applied multiple machine learning algorithms to perform transportation mode classification, and found that Random Forest [28] gave accurate classification.

Instead of using only GPS data or only accelerometer data, Ellis et al. [26] applied the Random Forest to *both GPS data and accelerometer data* in order to successfully perform transportation mode classification with a relatively high accuracy.

Other than using both GPS data and accelerometer data, Chung and Shalaby [27] developed a system that uses *both GPS and GIS data* instead. Their system classified four transportation modes—“walk”, “bike”, “bus” and “car”—for GPS-based travel surveys by using a rule-based algorithm and a map-matching algorithm [29] to detect the exact roads people moved on. However, the accuracy of the system is dependent on the corresponding GIS data. Similarly, Stenneth et al. [30] also used both GPS and GIS data when building their real-time transportation mode classification system. To perform the classification, they used the Random Forest as the supervised machine learning algorithm to identify a person’s current transportation mode.

3 Our Proposed Classification System

In this section, we describe our ground transportation mode classification system that uses GPS data, accelerometer data, GIS data, and/or window history queue. The system consists of the following five modules:

1. Dataset collection module,
2. Trip segmentation module,
3. Feature extraction module,
4. Model construction module, and
5. Data classification module.

The end result of the system are segmented trips (or trip windows), where each segment is labelled with the ground transportation mode the person used for the period of the segment. See Fig. 1 for the system layout. Figure 2 shows a zoom-in view of the dataset collection module (in which the MongoDB data store is highlighted in yellow) and the analysis component (highlighted in green), which consists of the last four modules (i.e., the trip segmentation, features extraction, model construction, and data classification modules) listed above. This data analysis component interacts with the dataset collection module.

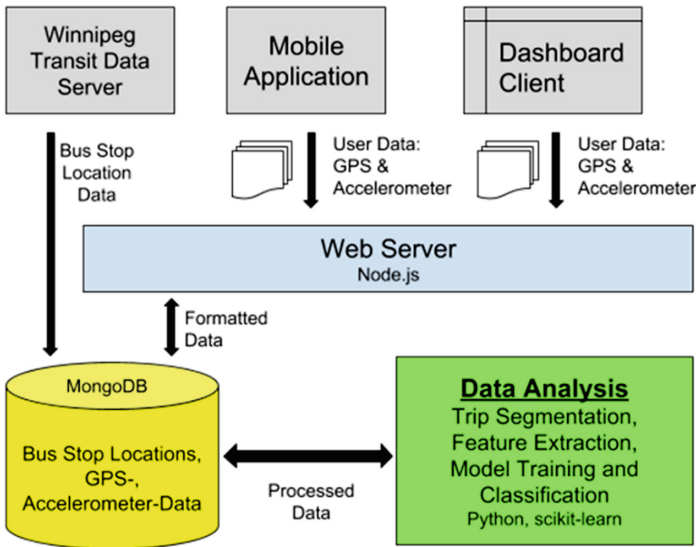


Fig. 1. Layout of our transportation mode classification system.

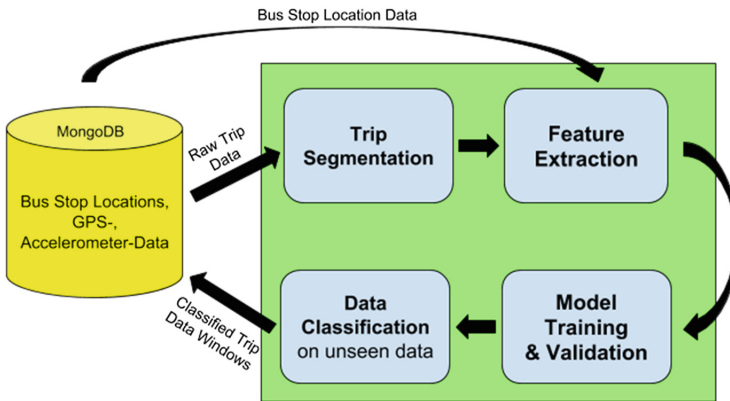


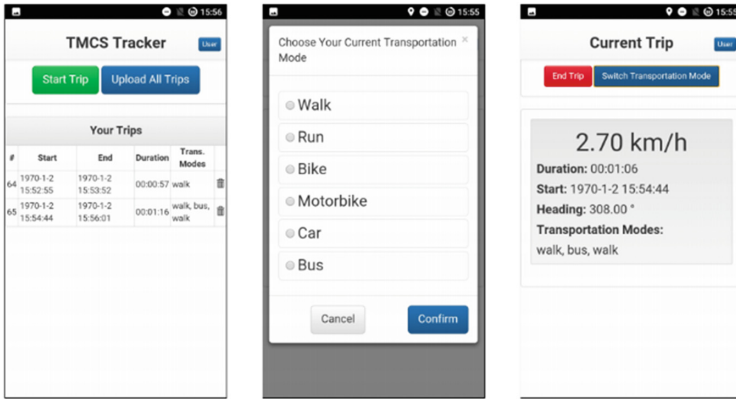
Fig. 2. Layout of the transportation data analysis component of our transportation mode classification system.

3.1 Dataset Collection Module

First, the *dataset collection module* collects trip traces (GPS locations) and trip accelerometer data, and stores them in a database (e.g., MongoDB). In addition, the module also collects the bus stop locations in a city—via its transit application programming interface (API)—when “bus” is one of the ground transportation modes for classification.

Figure 3 shows the main screens of an iOS application for *data collection*. Figure 3 (a) shows the “current trip” screen of the application, with which users can see their

current trip information (e.g., speed, alerts, map). To start a new trip or trip leg, the users simply end a trip. Then, a new trip will automatically start. After a new trip or trip leg starts, the users will be presented with a pop-up list of transportation modes, as shown in Fig. 3(b). Figure 3(c) shows the screen that lists users’ saved trip log, where they can review trip information.



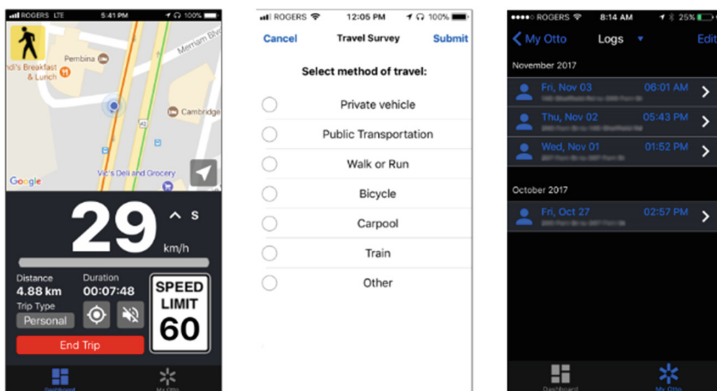
(a) The start screen and the user’s list of trips.

(b) Users can select a transportation mode at the beginning of a trip and during a trip when transitioning between different modes.

(c) Users can see some information relating to their current trip.

Fig. 3. Application for data collection.

The application also keeps track of users’ movement. Figure 4(a) shows the start screen of the application for *movement tracking*, with which users can manage their recorded trips, start new trip recordings and upload existing trips. Figure 4(b) shows



(a) The start screen and information relating to the user’s current trip.

(b) Users can select a transportation mode at the beginning of a trip. Trip legs are represented by individual trips.

(c) Users can see a list of their stored trips.

Fig. 4. Application for movement tracking.

the transportation mode selection pop-up that would appear when the users start a new trip. They could also open the popup anytime from the current trip screen when they are switching transportation modes. Figure 4(c) shows the screen for current trip information after the users started a new trip. Here, the GPS sampling rate was set to 1 Hz and the accelerometer sampling was at 22 Hz.

Recall that some related works use GIS data together with GPS location. Knowing the difficulty in obtaining a complete set of GIS information in some real-life situations, the only GIS information required by our data collection module is the bus stop location, which can be easily accessible. For example, in evaluation, we obtained the bus stop locations from Winnipeg Transit Open Data Web Service API.

3.2 Trip Segmentation Module

After collecting raw trip data by the data collection module, our *trip segmentation module* segments every trip (which is simply a collection of data points collected during a person’s entire commute from origin to destination—say, from home to work) based on the transportation mode used in each segment. For example, for a trip from home to work can be divided into the following three segments, as shown in Fig. 5:

1. Walk from home to the departure bus stop,
2. Bus from departure bus stop to destination bus stop, and
3. Walk from destination bus stop to office.

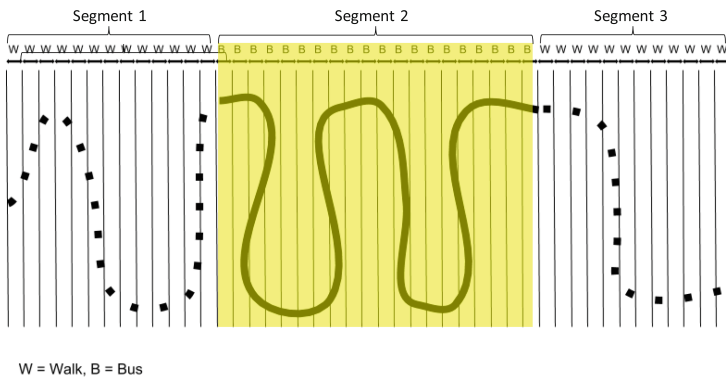


Fig. 5. Segmented trip.

Our trip segmentation module is designed in such a way that it automatically achieves the trip segmentation by simply classifying the transportation mode of each window of data. To elaborate, data of a trip are divided into many small windows of equal time interval. Moreover, when a transportation mode change occurs, data are assigned to different windows so that no two transportation modes are mixed within the same window. Segmenting the data into small windows led to a benefit that classification can be performed in real-time. For instance, as soon as sufficient amount of data

has been collected to fill a new window, the window can be classified with a transportation mode. Once every window is classified with a transportation mode, the user simply concatenates the windows/trip segments (each of which is labelled with a transportation mode) and presents each label on a map with a color-scheme for different transportation modes. Once the trip is rendered on a map, the user can easily identify different legs of the trip by simply looking at the different colors of the trip.

3.3 Feature Extraction Module

With (i) the bus stop location data (i.e., GIS information) collected by the dataset collection module and (ii) the GPS and accelerometer data associated with the trip segmented returned by the trip segmentation module, our *feature extraction module* extracts appropriate features for transportation mode classification. Specifically, the module extract the following three key types of features:

- **GIS-based features**, which capture the following GIS information related to bus stop locations:
 - a. Stopped at a bus stop, which is a Boolean feature that indicates whether or not a person stopped at any of the nearby bus stops within the window;
 - b. The number of bus stops, which captures the number of unique bus stops within the window;
 - c. The number of stops at bus stops, which captures the number of stops near all the bus stops within the window; and
 - d. Distance to closest bus stop (in meters), which captures the distance to the bus stop that is closest to the person within the entire window.
- **GPS-based features**, which capture the following geo-location and time information provided by GPS sensors:
 - a. Maximum speed (in km/h);
 - b. Average speed (in km/h);
 - c. Average altitude (in meters);
 - d. Average location accuracy (in meters);
 - e. Travel distance (in meters), which computes the geodesic distance (i.e., shortest possible line between two points p_1 and p_2 on a sphere) in terms of *Haversine distance*. Such a distance d between p_1 and p_2 can be calculated by:

$$d = 2r \sin^{-1} \left(\sqrt{\sin^2 \left(\frac{lat_2 - lat_1}{2} \right) + \cos(lat_1) \cos(lat_2) \sin^2 \left(\frac{long_2 - long_1}{2} \right)} \right)$$

where (i) r is the radius of the sphere, (ii) $long_1$ and lat_1 are respectively the longitude and latitude of p_1 , and (iii) $long_2$ and lat_2 are respectively the longitude and latitude of p_2 ; and

- f. GPS signal loss, which is a Boolean feature that indicates whether there is GPS signal or no signal.

- **Accelerometer-based features**, which capture the following measurement on acceleration of different transportation modes (e.g., automobile):
 - a. Maximum magnitude;
 - b. Minimum magnitude;
 - c. Average magnitude;
 - d. 25th percentile magnitude, which captures the average of all magnitude in the 25th percentile;
 - e. 75th percentile magnitude, which captures the average of all magnitude in the 75th percentile;
 - f. Magnitude standard deviation;
 - g. Lag-1 autocorrelation;
 - h. Correlation between x - and y -axes;
 - i. Correlation between x - and z -axes;
 - j. Correlation between y - and z -axes;
 - k. Average *roll*, which captures the average “bank angle” about rotations along the x -axis;
 - l. Average *pitch*, which captures the average “elevation” about rotations along the y -axis; and
 - m. Average *yaw*, which captures the average “bearing” about rotations along the z -axis.

Recall from Sect. 2 that some existing classifiers use only GPS based data (say, the aforementioned 6 GPS-based features), some use only accelerometer-based data (say, the aforementioned 13 accelerometer based features), some use both GPS and accelerometer based data (say, the aforementioned $6 + 13 = 19$ GPS- and accelerometer- based features), and some use both GPS and GIS based data (say, the aforementioned $6 + 4 = 10$ GPS- and GIS-based features). However, to the best of our knowledge, *no classifier—except our proposed system—uses all three types of $4 + 6 + 13 = 23$ features from these GIS-, GPS- and accelerometer-based data.*

To a further extent, with an aim to enhance the classification accuracy, we propose a novel concept of **window history queue (WHQ)**. The idea behind a WHQ is that previous data windows could help to classify the current data window. For instance, if the current data are similar to the previous one (e.g., similar average speed), then the current data are more likely to have the same classification as the previous data. Conversely, if the current data are quite different from the previous one, then there is a chance that transportation modes have changed. Our classifier is able to determine the subtle data differences during the training phase. For example, the random forest model can learn that a previous high average speed followed by a very low (current) average speed would mean that the current data represent the transportation mode of “walk”.

To support this concept of window history queue (WHQ), our feature extraction module extracts $6 + 4 = 10$ additional GPS- and accelerometer-based features for WHQ. These 10 additional features are listed as follows:

- **Additional GPS-based features for WHQ** include:
 - a. Previous maximum speed (in km/h), which is the maximum speed for all previous windows in the WHQ;

- b. Previous average speed (in km/h), which is the average speed for all previous windows in the WHQ;
 - c. Previous maximum average speed (in km/h), which is the maximum average speed for all previous windows in the WHQ;
 - d. Delta maximum speed (in km/h), which is the difference between the maximum speed of the current window and that of the previous windows;
 - e. Delta average speed (in km/h), which is the difference between the average speed of the current window and that of the previous windows; and
 - f. Delta maximum average speed (in km/h), which is the difference between the maximum average speed of the current window and that of the previous windows.
- **Additional accelerometer-based features for WHQ** include:
 - a. Previous average magnitude which captures the average of all average magnitudes in the WHQ;
 - b. Previous 25th percentile magnitude, which captures the average of all magnitude in the 25th percentile of the WHQ;
 - c. Previous 75th percentile magnitude, which captures the average of all magnitude in the 75th percentile of the WHQ; and
 - d. Previous magnitude standard deviation, which captures the average of all magnitude standard deviations of the WHQ.

3.4 Model Construction Module

Once features are extracted from GIS information, GPS sensors and accelerometer data, our *model construction module* builds, trains and validates a classification model. Specifically, it builds the random forest model by using the extracted features which are calculated based on the collected raw data. The extracted features are stored on a per-trip basis. For each trip, there is a set of feature windows. The trips for each transportation mode are shuffled and then split into two sets: (i) the training set and (ii) the testing/validation set. As a preview, to evaluate our classification system, we used 70% of the data for stratified 10-fold cross-validation with a 50-50 partition split between the training and the testing data for each partition in order to determine the accuracy of the random forest based classifier.

3.5 Data Classification Module

After constructing the classification model, our *data classification module* classifies unseen data and stores the classified trips back to the MongoDB. Specifically, segments of a trip are classified according to the ground transportation mode (e.g., “walk”, “bike”, “bus”, “car”) used by the user.

4 Evaluation

To evaluate our proposed ground transportation mode classification system, we conducted experiments on a computer running Ubuntu 16.04 LTS as the main operating system. The CPU was an AMD Phenom II X6 1100T with 6 cores clocked at 3.3 GHz to 3.7 GHz. There are 16 GB of RAM and a solid state drive in the computer.

We collected the GIS information (i.e., bus stop locations) from Winnipeg Transit Data Server by querying trip data via the Winnipeg Transit API. Users anonymously and securely uploaded their saved GPS- and accelerometer-based data via the mobile applications or dashboard. These trip information from users were then stored in a MongoDB, which supports basic geo-spatial query capabilities. The trip information was collected throughout a year, which contains trips with different weather and road conditions from summer to winter times. It captures the ground transportation mode (e.g., “walk”, “car”, “bus”) used by the user at the time of commute.

Recall from Sect. 3.2 that the trip segmentation module of our proposed ground transportation mode classification system divides each trip into many small windows of equal time interval. *Our first set of experiments is to determine an appropriate window size.* We varied the window size. The experimental results shown in Fig. 6 reveal that a window size of 4 s gave the most accurate classification.

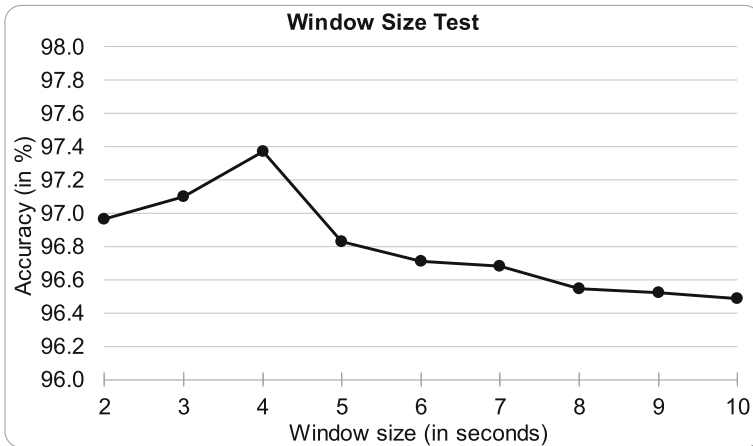


Fig. 6. Experimental result: Segmented window size.

Recall from Sect. 3.3 that the feature extraction module of our proposed ground transportation mode classification system uses a novel concept of window history queue (WHQ), which captures historical data. With WHQ, comparisons can then be made between the GPS information or accelerometer data in the current window and those in the previous windows within the WHQ. So, *our second set of experiments is to determine an appropriate queue length.* We varied the queue length. The experimental results shown in Fig. 7 reveals that a WHQ length of 15 windows (each window of size 4-second interval) gave the most accurate classification.

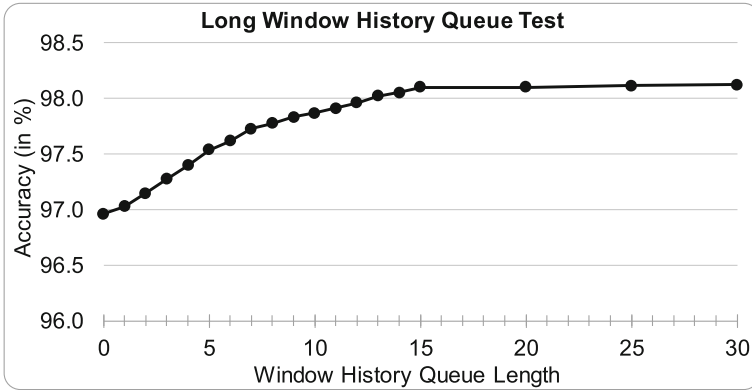


Fig. 7. Experimental result: Window history queue (WHQ) length.

To measure the effectiveness of our proposed ground transportation mode classification system, we use the standard measures of precision and recall. *Precision* measures the positive predictive/classified value, i.e., the fraction of true positives among all positives (i.e., true and false positives):

$$\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positive}}$$

Recall measures the true positive rate or sensitivity, i.e., the fraction of true positives among true positives and false negatives:

$$\text{Recall} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}}$$

Accuracy measures the fraction of true positives and true negatives among all predications/classifications (i.e., among all positives and negatives):

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

where TP denotes true positives, TN denotes true negatives, FP denotes false positives, and FN denotes false negatives. So, *our third set of experiments is to compute the precision, recall, and accuracy of the classification results on unseen data*. The experimental results show that our system accurately classified different segments of unseen data with highly accurate (e.g., above 95%) ground transportation modes.

Recall from Sects. 2 and 3 that some existing classifiers use only GPS-based data (say, the 6 GPS-based features), some use only accelerometer-based data (say, the 13 accelerometer-based features), some use both GPS- and accelerometer-based data (say, the 6 + 13 = 19 GPS- and accelerometer-based features), and some use both GPS- and GIS-based data (say, the 6 + 4 = 10 GPS- and GIS-based features). In contrast, *our ground transportation mode classification system uses all GPS-, GIS- and*

accelerometer-based features, which is a key contribution of this paper. Hence, our fourth set of experiments is to compare the accuracy of the classification results of our classification system with related works. The experimental results shown in Table 1 reveals that our system led to higher classification accuracy.

Table 1. Classification accuracy of our classifier with compared with related works.

	Data used	Accuracy
Related Works	Only GPS data	88.87%
	GPS + GIS data	88.96%
	Only accelerometer (Acc) data	93.13%
Our proposed classification system	GPS + Acc data	95.58%
	GPS + GIS + Acc data	96.97%

Recall from Sect. 3 that *another key contribution of this paper is our proposal of the concept of window history queue (WHQ).* Hence, our fifth set of experiments is to measure the benefits (e.g., increase in classification accuracy) of using WHQ. The experimental results shown in Table 2 reveals that our system (which uses the WHQ) led to higher classification accuracy.

Table 2. Classification accuracy of not using vs. using the WHQ.

Without WHQ	Accuracy	With WHQ	Accuracy
Only GPS data	88.87%	GPS + WHQ	94.70%
GPS + GIS data	88.96%	GPS + GIS + WHQ	94.63%
Accelerometer (Acc) data	93.13%	Acc + WHQ	94.85%
GPS + Acc data	95.58%	GPS + Acc + WHQ	98.08%
GPS + GIS + Acc data	96.97%	GPS + GIS + Acc + WHQ	98.09%

5 Conclusions

This paper focuses on urban data mining. In particular, we proposed a ground transportation mode classification system. The system consists of five modules. Among them, the dataset collection module collects the GIS information about bus stop locations, the geo-location and time information provided by GPS sensors, and accelerometer-based data capturing the acceleration of different transportation modes. These data are collected from different sources: GIS information from the city or transit company, whereas GPS- and accelerometer-based data are collected by users. Then, the trip segmentation module segments data about a trip into many windows of size 4 s, and each of these windows captures a single mode of transportation. Afterwards, the feature extraction module extracts the standard GIS-, GPS- and accelerometer-based data. Most, if not all, of the existing related works do not use all GIS-, GPS- and accelerometer-based data. So, we proposed in this paper to use the combined GIS-, GPS- and accelerometer-based data. Consequently, the model construction module

builds and trains a random forest classification model using these combined data. The data classification module then classifies future unseen data. Experimental results show that the use of combined GIS-, GPS- and accelerometer-based data led to high classification accuracy.

Moreover, with an aim to improve classification accuracy, we also proposed in this paper a novel concept of window history queue (WHQ) so that the model construction module can compare the data in the current window with data in the previous windows within the WHQ. To facilitate our proposed concept of WHQ, the feature extraction module extracts additional GPS- and accelerometer-based data for WHQ. Experimental results show that the use of WHQ led to higher accuracy than their counterparts that do not use the WHQ. These results demonstrate the effectiveness of our system in classifying ground transportation modes for urban data mining in smart cities.

As ongoing and future work, we are examining relationships (e.g., correlations) among the extracted features to see if the number of features can be reduced. At the same time, we are also exploring new features that could further enhance the classification accuracy. Moreover, we are also conducting more exhaustive evaluation (e.g., comparisons with fitness trackers).

Acknowledgements. This project is partially supported by NSERC (Canada) and University of Manitoba. Thanks F. Franczyk—President of Presen Technologies Inc. (PERSENTECH)—for his introduction of OttoFleet Mobile™ (an application for GPS-enabled iPhones®), which inspired the design of the dataset collection module of our classification system.

References

1. Braun, P., Cuzzocrea, A., Jiang, F., Leung, C.K.-S., Pazdor, A.G.M.: MapReduce-based complex big data analytics over uncertain and imprecise social networks. In: Bellatreche, L., Chakravarthy, S. (eds.) DaWaK 2017. LNCS, vol. 10440, pp. 130–145. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-64283-3_10
2. Chen, Y.C., Wang, E.T., Chen, A.L.P.: Mining user trajectories from smartphone data considering data uncertainty. In: Madria, S., Hara, T. (eds.) DaWaK 2016. LNCS, vol. 9829, pp. 51–67. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-43946-4_4
3. Hoi, C.S.H., et al.: Supporting social information discovery from big uncertain social key-value data via graph-like metaphors. In: Xiao, J., Mao, Z.-H., Suzumura, T., Zhang, L.-J. (eds.) ICCM 2018. LNCS, vol. 10971, pp. 102–116. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94307-7_8
4. Egho, E., et al.: MiSeRe-Hadoop: a large-scale robust sequential classification rules mining framework. In: Bellatreche, L., Chakravarthy, S. (eds.) DaWaK 2017. LNCS, vol. 10440, pp. 105–119. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-64283-3_8
5. Leung, C.K.: Big data analysis and mining. In: Encyclopedia of Information Science and Technology, 4th edn., pp. 338–348 (2018)
6. Leung, C.K., Jiang, F., Pazdor, A.G.M., Peddle, A.M.: Parallel social network mining for interesting ‘following’ patterns. *Concurr. Comput. Pract. Exp.* **28**(15), 3994–4012 (2016)
7. Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P.: From data mining to knowledge discovery: an overview. In: *Advances in Knowledge Discovery and Data Mining*, pp. 1–34 (1996)

8. Behnisch, M., Ultsch, A.: Urban data mining using emergent SOM. In: Preisach, C., Burkhardt, H., Schmidt-Thieme, L., Decker, R. (eds.) *Data Analysis, Machine Learning and Applications. Studies in Classification, Data Analysis, and Knowledge Organization*, pp. 311–318. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78246-9_37
9. Andrienko, G., et al.: Mining urban data (Part A). *Inf. Syst.* **54**, 113–114 (2015)
10. Andrienko, G., et al.: Mining urban data (Part B). *Inf. Syst.* **57**, 75–76 (2016)
11. Andrienko, G., et al.: Mining urban data (Part C). *Inf. Syst.* **64**, 219–220 (2017)
12. Sokmenoglu, A., Cagdas, G., Sariyildiz, S.: Exploring the patterns and relationships of urban attributes by data mining. In: *eCAADe 2010*, pp. 873–881 (2010)
13. Murakami, E., Wagner, D.P., Neumeister, D.M.: Using global positioning systems and personal digital assistants for personal travel surveys in the United States. In: *Transport Surveys: Raising the Standard*, art. III-B (2000)
14. Ettema, D., Timmermans, H., van Veghel, L.: *Effects of Data Collection Methods in Travel and Activity Research* (1996)
15. Stopher, P.R.: Household travel surveys: cutting-edge concepts for the next century. In: *Conference on Household Travel Surveys*, pp. 11–23 (1995)
16. Maat, K., Timmermans, H.J.P., Molin, E.: A model of spatial structure, activity participation and travel behavior. In: *WCTR 2004* (2004)
17. Stopher, P.R.: Use of an activity-based diary to collect household travel data. *Transportation* **19**(2), 159–176 (1992)
18. Schlich, R., Axhausen, K.W.: Habitual travel behaviour: evidence from a six-week travel diary. *Transportation* **30**(1), 13–36 (2003)
19. Arentze, T., et al.: New activity diary format: design and limited empirical evidence. *TRR* **1768**, 79–88 (2001)
20. Forrest, T., Pearson, D.: Comparison of trip determination methods in household travel surveys enhanced by a global positioning system. *TRR* **1917**, 63–71 (2005)
21. Wolf, J., Guensler, R., Bachman, W.: Elimination of the travel diary: experiment to derive trip purpose from global positioning system travel data. *TRR* **1768**, 125–134 (2001)
22. Biljecki, F., Ledoux, H., van Oosterom, P.: Transportation mode-based segmentation and classification of movement trajectories. *IJGIS* **27**(2), 385–407 (2013)
23. Zheng, Y., Chen, Y., Li, Q., Xie, X., Ma, W.: Understanding transportation modes based on GPS data for web applications. *ACM TWeb* **4**(1), art. 1 (2010)
24. Hemminki, S., Nurmi, P., Tarkoma, S.: Accelerometer-based transportation mode detection on smartphones. In: *SenSys 2013*, art. 13 (2013)
25. Shaque, M.A., Hato, E.: Use of acceleration data for transportation mode prediction. *Transportation* **42**(1), 163–188 (2015)
26. Ellis, K., et al.: Identifying active travel behaviors in challenging environments using GPS, accelerometers, and machine learning algorithms. *Front. Pub. Health* **2**, art. 36 (2014)
27. Chung, E., Shalaby, A.: A trip reconstruction tool for GPS-based personal travel surveys. *Transp. Plann. Technol.* **28**(5), 381–401 (2005)
28. Breiman, L.: Random forests. *Mach. Learn.* **45**(1), 5–32 (2001)
29. Greenfeld, J.: Matching GPS observations to locations on a digital map. In: *TRB 81st Annual Meeting* (2002)
30. Stenneth, L., Wolfson, O., Yu, P.S., Xu, B.: Transportation mode detection using mobile phones and GIS information. In: *ACM SIGSPATIAL GIS 2011*, pp. 54–63 (2011)



Location Prediction Using Sentiments of Twitter Users

Ritu Singh^(✉) and Durga Toshniwal

Department of Computer Science and Engineering,
IIT Roorkee, Roorkee 247667, India
ritusingh2081992@gmail.com

Abstract. This study aims to predict the next location of a twitter user only by using his past tweets. Twitter is a very popular micro-blogging platform and a lot of people tweet about different topics varying from personal day-to-day activities to some global event. This provides us with the opportunity to perform sentiment analysis on their past tweets for prediction of their next visit. Sentiment analysis helps in revealing the opinion, desire or intentions of a person looking at the text that they write. In this paper, a new model called Sentiments based Labeled LDA model (SLLDA) is proposed to predict users' next location within a given geo-spatial range. This kind of prediction can be used by various establishment owners for the targeted promotions of their products. This can also be helpful for personalized recommendation. Various experiments have been performed to evaluate the performance of the proposed model. The proposed model outperforms in every set of experiments and is better than each baseline model considered in the study. The accuracy comparison has also been done for different window lengths of past tweets and different radii of query. The performance of the proposed model turned out to be better for each set of experiments.

Keywords: Location prediction · Sentiment analysis · Topic modelling
Twitter

1 Introduction

With the increase in trend of using various social media platforms, humongous amount of unstructured textual and pictorial data is generated daily. Various micro-blogs have recently captured peoples' attention where they can share their interests and opinions via short posts [1–3]. One of the most famous and widely used micro-blogs is Twitter. People from all around the world use Twitter frequently to share the details about their day-to-day activities or to express their views on some social event using tweets.

Sentiment analysis or opinion mining is also one of the trending techniques in the field of data analysis to classify the user-generated text into positive, negative, neutral or more sentimental classes. Sentiment analysis is generally applied over the user-generated text like reviews, surveys, social media posts etc. It has been used widely to predict election results, stock market etc. However, no work has been done to predict the next location of users by analyzing their sentiments.

The popularity of twitter has led to huge amount of raw data which can be used in many applications like prediction of stock market [4] and election results, friend recommendation, location prediction, home location identification etc. This study deals with the prediction of next location that a user might visit by seeing only his past tweets. The location here means the category of location i.e. restaurant, gym, pub etc. Various establishment owners for their promotional activities can use this kind of prediction. They can focus and send their promotional messages only to a group of people who are likely to visit their kind of business in their proximity in near future, thus saving a lot of money. Knowing someone's location beforehand can also be used by many recommendation systems like location or movie recommendation. This can also be helpful for the common users in filtering out the best available options for them, as they would be getting personalized recommendations and targeted promotions based on their future interests.

In this paper, next visit of a twitter user at a given time and within a given geospatial range is predicted by combining topic modeling and sentiment analysis techniques. First, the past tweets of the user has been analyzed sentimentally to get a glimpse of what the user actually wants to say and then to predict the next visit of the user, topic modeling and sentiment analysis has been combined. For example, if a user writes I am not hungry, then topic modeling alone might predict the next visit of the user to be a restaurant by focusing on the word hungry. However, what user actually wants to say here is completely different.

The main contributions of this paper are:

- Proposal of a new model that performs topic-sensitive sentiment analysis for the next location prediction.
- Performance comparison of the proposed approach on different set of experiments.

The rest of the paper is organized as: Sect. 2 describes previous related work in the field of sentiment analysis and location prediction. Section 3 presents in detail the proposed model for the prediction of next visit of a twitter user. Section 4 talks about the dataset used, various experiments, and their results obtained. Finally, Sect. 5 concludes the paper.

2 Related Works

Sentiment analysis is a trending technique in the field of data mining and data analysis. Speriosu et al. [5] applied a label propagation method for sentiment analysis on twitter. The proposed approach leveraged the follower graph of twitter. Tweets, users, hashtags, unigrams, bigrams and emoticons were used as the nodes for constructing the graph.

Cui et al. [6] came up with an algorithm for sentiment analysis on twitter. The algorithm was based on label propagation by analyzing the emotion tokens. Wang et al. [7], instead of using emotion tokens, proposed a model based on graphs that leveraged co-occurrence of the hashtags to find out the sentiment of other hashtags.

The other sentiment analysis technique is lexicon based which uses external lexicon, like SentiWordNet [8], LIWC lexicon [9] etc. to train data. Thelwall et al. [10, 11]

proposed sentiStrength sentiment analysis technique that applies various lexical rules to the sentences to identify its sentiment. Lexical rules adopted by the algorithm are spelling correction algorithm and consideration of booster words to increase or reduce the sentiment strength of the subsequent words. The major drawback of this technique is that it can't be effectively applied to tweets as the word list it relies on is very small in length.

Ortega et al. [12] proposed a three step unsupervised sentiment analysis technique in which preprocessed tweets are detected for polarity. A rule-based classifier was applied as the last step. The classifier and polarity detection were based on Senti-WordNet and WordNet. SentiCircle [13] proposed by Hassan Saif et al. captures the contextual sentiment of a word and hence the tweet. A word's actual sentiment is expressed by the context in which it is talked about. To calculate the overall sentiment of a word, they considered all the context words of the main word and made a term-context vector out of it.

Mathew et al. [14] predict the movement of an individual by clustering the location histories visited by an individual according to the locations' characteristics i.e. the time in which they were visited. They train the Hidden Markov Model for each such cluster formed and hence capture the sequential relations between places visited in given time. Bao et al. [15] utilized the growing popularity of location based social networks in understanding a user's preferences based on his location history. They presented a recommendation system that is location-based and is aware of the user's preference.

Yuan et al. [16] proposed a point-of-interest (POI) recommendation system based on geographical and temporal influences aware graph to model geographical influences, check-in records and the corresponding temporal influences. The approach proposed by them works only if the location histories are available for the users.

Arun et al. in [17] put forward a novel approach to predict the next location visited by twitter users only by looking at their past tweets. They considered two feature sets in order to do so. First, one was the personality traits of the user and second one being the users' past tweets. They applied a famous topic modeling technique, Labeled LDA [18], along with time factor on the past tweets of user. However, their major drawback was not considering the mood or opinion users expressed through their tweets. This gave us the opportunity to also consider the sentiments of a person towards a particular place type by looking at their tweets and then make the predictions.

However, the technique proposed in this study is based on a famous topic modeling approach, Labeled LDA (LLDA). It integrates sentiment analysis and LLDA for the purpose of topic modeling i.e. predicting the next visit of a twitter user.

3 Proposed Approach

3.1 Label Tweets with Location Categories

The raw tweets obtained from Twitter are not labeled with the location information. So to label the tweets with location information, all the tweets having location information i.e. geo-coordinates and location indicative words like @, I am at, I am @, I'm at, I'm @, i am at, i'm at, i'm @ are filtered out.

If similarity, as given by Eq. (1), between three words written after the location indicative words and place names returned by Google Places API [19] is greater than or equal to 75% then the corresponding tweet is labeled with the location category information of the matched entry of GPA.

$$Sim(w_1, w_2) = \frac{|w_1 \cap w_2|}{|w_2|} \tag{1}$$

Where $|w_1 \cap w_2|$ is the number of words common between w_1 and w_2 , $|w_2|$ is the number of words in w_2 .

3.2 Build Training and Test Datasets

The next visit of a user is predicted here by looking at his past tweets. For this, users are divided into two groups, one with greater than 60 location tweets in their timeline and rest of the users are put in other group. Training dataset is made from all the location tweets (except latest 50) of the users belonging to group-1. Test dataset-1 is made from rest of the tweets i.e. latest 50 location tweets of group-1 users and test dataset-2 is completely made from users belonging to group-2.

3.3 Sentiment Analysis

The sentiment analysis technique applied in this study is based on the Contextual semantics for sentiment analysis done by Hassan et al. in [13].

The overall sentiment analysis technique is explained in Fig. 1. For context dependent sentiment analysis, first term-context vector is generated from the whole corpus which is the number of times each context word appears with a particular word. Then prior sentiment scores are calculated for each word with the help of an external sentiment lexicon, SentiWordNet. A list of negation words is also used to invert the prior sentiment score of the word. Then term degree of correlation (TDOC) scores are calculated for each word in the tweet and its context terms present in the tweet, as done in [13]. Finally, to reassign the sentiment related to a term, median of its context terms present in that tweets is identified.

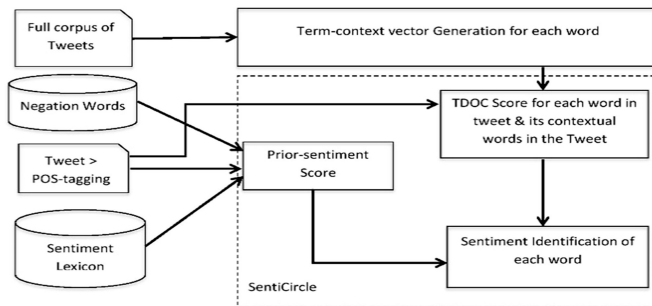


Fig. 1. Sentiment analysis workflow

3.4 Location Prediction Using SLLDA

The purpose of this study is to predict the next most probable location to be visited by a twitter user. The number of such locations is limited i.e. 99 in our case. Therefore, the model developed in this study i.e. Sentiments based Labeled LDA model is based on a famous topic modeling technique called Labeled LDA. In LLDA, there's a multinomial distribution of each label over all words and multinomial distribution of each label over a document. Also, each word in the document is derived by the preference of document for a label and how much a label prefers the word. Here, labels are the categories of the location to be predicted.

This study proposes a new technique for location prediction that is based on LLDA and sentiment analysis discussed in previous part. Assume D is the set of documents i.e. $D = \{d_1, d_2, \dots, d_M\}$ and each document is a set of words that comes from vocabulary V of distinct terms from the whole corpus. To learn new topics for each document and the words present in it, following steps are carried out:

- Choose some fixed number of topics.
- For each document d , randomly assign each word present in the document to one of the topics.
- Assign sentiment score to each word based on the contextual sentiment analysis technique, discussed in previous section.
- For each word w_i in the document d and each topic z_i , compute: $P(z_i | d)$, $P(w_i | z_i, l)$.
- Reassign the topic z_i to word w_i with the probability of $P(z_i | d) * P(w_i | z_i, l)$.

During estimation, the topic-word-sentiment distribution for a particular word is obtained by considering different topics assigned to it at different places and the sentiment score of that word at those places. Thus, in each iteration a sentiment threshold, range of sentiment values for each topic-word combination is obtained for a particular word-topic combination. If the sentiment score of the word at a particular position lies within the range of the threshold in the next iteration, then its sentiment value at the place is considered to be positive or neutral else negative. During inference, the threshold is obtained by looking at the training set.

A new topic is assigned to a word w as per the Eq. (2).

$$P(T) = P(z|d)P(w|z, l) = \frac{P(z|d)P(w|z)P(w|l)}{P(w)} \quad (2)$$

Where, w is the word under consideration, z is the topic assigned to it and l is the sentiment label and d is the document.

$$P(w) = \frac{N_w}{\sum_{j \in V} N_j} \quad (3)$$

Where N_w is the number of times w appears in the corpus.

The first term indicates the chances of new topic z to belong to the document d at position w , as per Eq. (4).

$$P(z|d) = \frac{\text{Number of words assigned to topic } z \text{ in doc } d}{\text{Total number of words in doc } d} \quad (4)$$

Second term comes from the word-topic distribution given by Eq. (5).

$$P(w|z) = \frac{\text{Number of instances where } w \text{ is assigned to } z}{\text{Total words assigned to topic } z} \quad (5)$$

Third term, defined by Eq. (6), is where the word-topic-sentiment distribution comes into play. Number of sentiment labels considered is three i.e. positive, negative and neutral.

$$P(w|l) = \frac{\text{Number of words assigned label } l}{\text{Total words assigned label } l} \quad (6)$$

3.5 Additional Constraints

Additional constraints considered here is the query radius. The intuition behind this constraint is that people tend to travel short distances more.

4 Experimental Results

4.1 Dataset Description

We crawled the publicly available Twitter data using Twitter Search API [20]. The dataset contains tweets of randomly chosen 4,606 twitter users belonging to New York City who tweet at least 20 times a day collecting approximately 3,200 tweets from each users' timeline. In the dataset, out of 1,05,28,618 total tweets, only 21.34% of tweets are geo-tagged.

Total 99 location categories were found in our dataset, including subway_station, department_store, embassy etc. Out of 4,606 users, 1331 users were categorized in dataset_1 and rest of the users (3275) were put in dataset_2. Also, 2,12,219 tweets were found in dataset_1 and 59,339 tweets belonged to dataset_2. Total number of tweets in training dataset were 1,50,749 and size of test dataset_1 was 61,470 and that of test dataset_2 was 59,339.

4.2 Experiments and Results

We have conducted three sets of experiments to show the effectiveness of our proposed model. Accuracy is measured as per Eq. (7).

$$\text{Accuracy} = \frac{\text{Number of instances correctly predicted}}{\text{Total number of test instances}} \quad (7)$$

4.2.1 Comparison with Baseline Model

The baseline models considered for the first set of experiments are:

- Labeled LDA.
- Baseline-1: Nearest distance model.
- Baseline-2: Google popularity model.

Figures 2 and 3 compare the results of various baseline models considered in this study. It can be seen from the Figs. 2 and 3 that the proposed approach outperformed the various baseline models. Figure 3 also shows that the proposed approach is better than the baseline models for the users that are not shown to the model during its training phase.

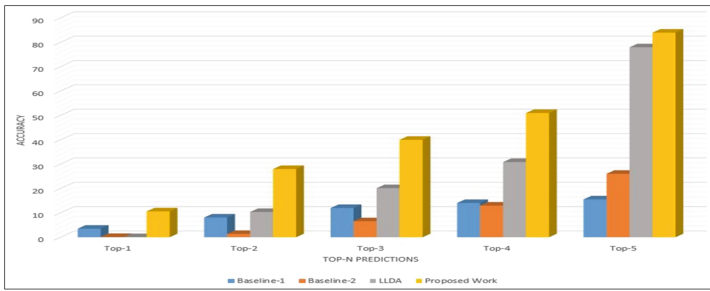


Fig. 2. Comparison of baseline models with proposed model for test dataset_1

4.2.2 Comparison with Different Window Sizes

Past tweets of a user are taken with window slots of 24 h, 12 h, 6 h and 3 h. Initially, experiments were conducted by taking a window slot of 24 h. Figure 4 shows the performance results of the proposed approach for this window slot.

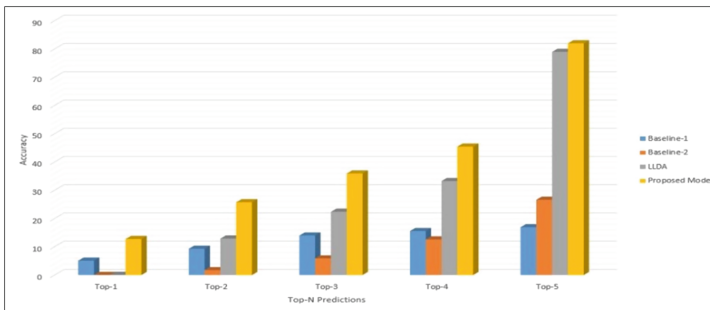


Fig. 3. Comparison of baseline models with the proposed model for test dataset_2

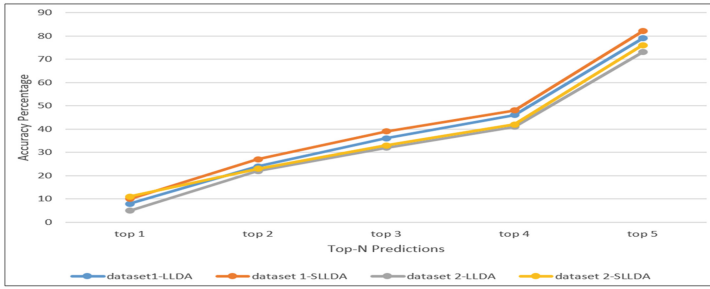


Fig. 4. LLDA vs Proposed model for window size of 24 h

To see the performance of the models in a more specific and precise environment, the size of the window slot was changed to 12 h. Figure 5 shows the comparison of the models for the past 12 h tweets.

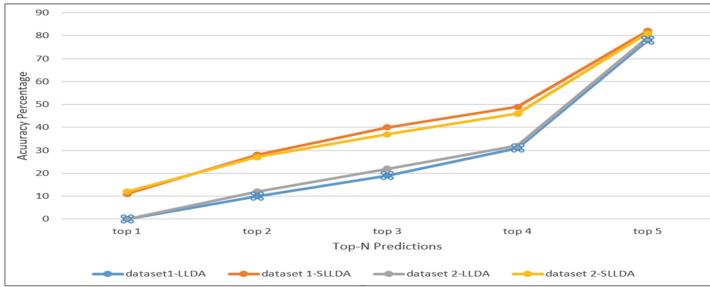


Fig. 5. LLDA vs Proposed model for window size of 12 h

Figures 6 and 7 shows the performance variations of LLDA and proposed approach for the past 6 h and 3 h tweets respectively.

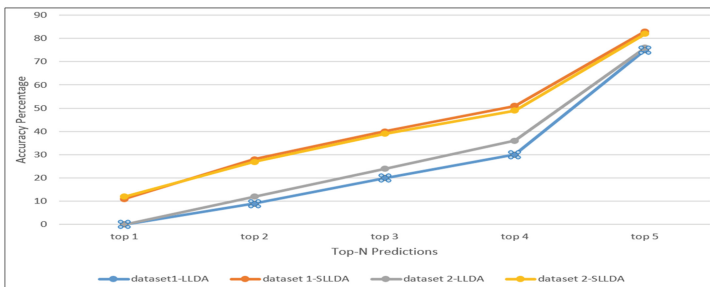


Fig. 6. LLDA vs Proposed model for window size of 6 h

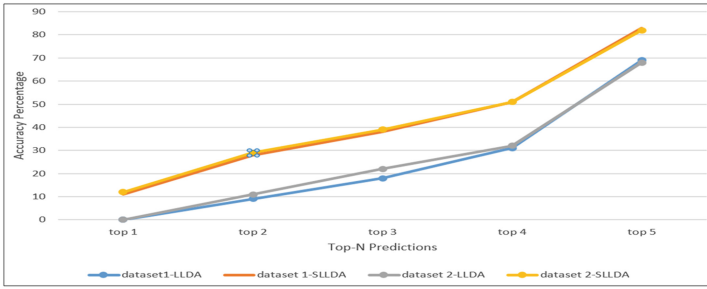


Fig. 7. LLDA vs Proposed model for window size of 3 h

It can be seen from the results of these figures that the proposed model is better than LLDA at any time slot. As the window size of the past tweets decreases, the accuracy difference between the proposed model and the LLDA model increases.

4.2.3 Comparison for Different Radii

To see the effect of query radius, experiments are done on the places at different distances of 300 m, 1000 m and 2000 m from the query point for the window size of past 12 h tweets.

Figure 8 compares the LLDA model and the proposed model for different radii. As it can be seen from the graph that the proposed model performs better than LLDA at different radii too. This shows that including sentiment analysis on the peoples’ tweets actually helps in getting predictions that are more accurate.

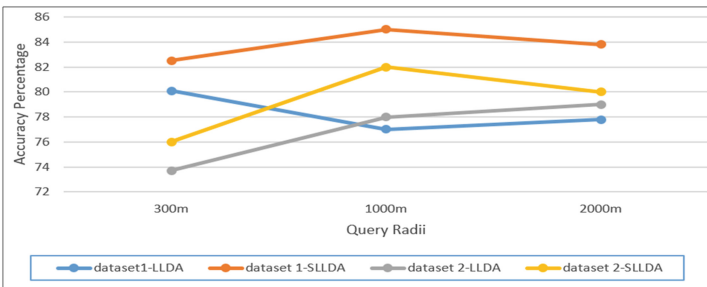


Fig. 8. LLDA vs Proposed model for different radii

5 Conclusion

This study is dedicated to predict the next location of a user within a geo-spatial range. The location considered here are the location categories like restaurant, shopping complex etc. In past, researchers have solved approximately the same problem using either just the text of tweets or by analyzing the user behavior. However, none have considered to use the sentiment of the words.

Therefore, this study extends the previous topic modeling technique by combining it with sentiment analysis model. It proposes a new topic modeling technique, SLLDA, which predicts the next place of visit based on the word-topic sentiment distribution. Various set of experiments were performed to compare the accuracy of the proposed model with the existing models. All the experiments proved that the proposed model is better than the existing ones.

One of the immediate extensions of the proposed model can be to consider the time of visit of a particular place. Not each place is as popular as the other at the same time.

References

1. Bhattacharya, P., Zafar, M.B., Ganguly, N., Ghosh, S., Gummadi, K.P.: Inferring ser interests in the Twitter social network. In: Kobsa, A., Zhou, M.X., Ester, M., Koren, Y. (eds.) Eighth ACM Conference on Recommender Systems, RecSys 2014, Foster City, Silicon Valley, CA, USA, 06–10 October 2014. ACM (2014)
2. Bollen, J., Mao, H., Zeng, X.-J.: Twitter mood predicts the stock market. *J. Comput. Sci.* **2**(1), 18 (2011)
3. Budak, C., Kannan, A., Agrawal, R., Pedersen, J.: Inferring user interests from microblogs. Technical report MSR-TR-2014-68 (2014). <http://research.microsoft.com/apps/pubs/default.aspx?id=217311>
4. Li, Q., Zhou, B., Liu, Q.: Can Twitter posts predict stock behavior? a study of stock market with Twitter social emotion. In: 2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), Chengdu, pp. 359–364 (2016)
5. Speriosu, M., Sudan, N., Upadhyay, S., Baldrige, J.: Twitter polarity classification with label propagation over lexical links and the follower graph. In: Proceedings of the First Workshop on Unsupervised Learning in NLP (EMNLP 2011), pp. 53–63. Association for Computational Linguistics, Stroudsburg (2011)
6. Cui, A., Zhang, M., Liu, Y., Ma, S.: Emotion tokens: bridging the gap among multilingual Twitter sentiment analysis. In: Salem, M.V.M., Shaalan, K., Oroumchian, F., Shakery, A., Khelalfa, H. (eds.) AIRS 2011. LNCS, vol. 7097, pp. 238–249. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25631-8_22
7. Wang, X., Wei, F., Liu, X., Zhou, M., Zhang, M.: Topic sentiment analysis in Twitter: a graph-based hashtag sentiment classification approach. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM 2011), pp. 1031–1040. ACM, New York (2011)
8. Baccianella, S., Esuli, A., Sebastiani, F.: Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In: Seventh Conference on International Language Resources and Evaluation, Malta. Retrieved May, Valletta, Malta (2010)
9. Pennebaker, J.W., Mehl, M.R., Niederhoffer, K.G.: Psychological aspects of natural language use: our words, our selves. *Annu. Rev. Psychol.* **54**(1), 547–577 (2003)
10. Thelwall, M., Buckley, K., Paltoglou, G.: Sentiment strength detection for the social web. *J. Am. Soc. Inf. Sci. Technol.* **63**(1), 163–173 (2012)
11. Thelwall, M., Buckley, K., Paltoglou, G., Cai, D., Kappas, A.: entiment strength detection in short informal text. *J. Am. Soc. Inf. Sci. Technol.* **61**(12), 2544–2558 (2010)

12. Ortega, R., Fonseca, A., Montoyo, A.: SSA-UO: unsupervised Twitter sentiment analysis. In: Proceedings of the 7th International Workshop on Semantic Evaluation - 2nd Joint Conference on Lexical and Computational Semantics (SemEval 2013), pp. 501– 507. Association for Computational Linguistics
13. Saif, Hassan, He, Yulan, Fernandez, Miriam, Alani, Harith: Contextual semantics for sentiment analysis of Twitter. *Inf. Process. Manag. Int. J.* **52**(1), 5–19 (2016)
14. Mathew, W., Raposo, R., Martins, B.: Predicting future locations with hidden Markov models. In: Dey, A.K., Chu, H.-H., Hayes, G.R. (eds.) The 2012 ACM Conference on Ubiquitous Computing, UbiComp 2012, Pittsburgh, PA, USA, 5–8 September 2012, pp. 911–918. ACM (2012)
15. Bao, J., Zheng, Y., Mokbel, M.F.: Location-based and preference-aware recommendation using sparse geo-social networking data. In: Proceedings of the 20th International Conference on Advances in Geographic Information Systems (SIGSPATIAL 2012), pp. 199–208. ACM, New York (2012)
16. Yuan, Q., Cong, G., Sun, A.: Graph-based Point-of-interest recommendation with geographical and temporal influences. In: Li, J., Wang, X.S., Garofalakis, M.N., Soboroff, I., Suel, T., Wang, M. (eds.) Proceedings of the 23rd ACM International Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, 3–7 November 2014, pp 659–668. ACM (2014)
17. Chauhan, A., Kummamuru, K., Toshniwal, D.: Prediction of places of visit using tweets. *Knowl. Inf. Syst.* (2016). <https://doi.org/10.1007/s10115-016-0936-x>
18. Ramage, D., Hall David, L.W., Nallapati, R., Manning, C.D.: Labeled LDA: a supervised topic model for credit attribution in multi-labeled corpora. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009, 6–7 August 2009, Singapore, A Meeting of SIGDAT, A Special Interest Group of the ACL, pp. 248–256. ACL (2009)
19. GoogleAPI (2015) Google Places API. <https://developers.google.com/places/documentation>
20. TwAPI (2015) Twitter streaming API. <https://dev.twitter.com/docs/using-search>

Classification and Clustering



A Clustering Model for Uncertain Preferences Based on Belief Functions

Yiru Zhang^(✉), Tassadit Bouadi, and Arnaud Martin

Univ Rennes 1, CNRS, IRISA, Rennes, France
{yiru.zhang,tassadit.bouadi,arnaud.martin}@irisa.fr
<http://www-druid.irisa.fr/>

Abstract. Community detection is a popular topic in network science field. In social network analysis, preference is often applied as an attribute for individuals' representation. In some cases, uncertain and imprecise preferences may appear. Moreover, conflicting preferences can arise from multiple sources. From a model for imperfect preferences we proposed earlier, we study the clustering quality in case of perfect preferences as well as imperfect ones based on weak orders (orders that are complete, reflexive and transitive). The model for uncertain preferences is based on the theory of belief functions with an appropriate dissimilarity measure when performing the clustering steps. To evaluate the quality of clustering results, we used Adjusted Rand Index (ARI) and silhouette score on synthetic data as well as on Sushi preference data set collected from real world. The results show that our model has an equivalent quality with traditional preference representations for certain cases while it has better quality confronting imperfect cases.

Keywords: Clustering for orders · Imperfect preference modeling
Theory of belief functions

1 Introduction

Community detection is a very popular topic in network science field, and has received a great deal of attention. It is a key task for identifying groups (*i.e. clusters*) of objects that share common properties and/or interact with each other. Many algorithms have been developed for efficient community detection. Depending on the information source used to perform the *clustering* task, these algorithms can be classified into two main categories: graph structure based techniques [12], and node attribute based techniques [18]. The first one considers the *relationships* and the *connections* between the objects (*e.g.* friendships or professional relationships between social network agents¹, proteins interactions, etc.), while the second one analyses the similarity between objects based

¹ In decision making theory, different terms may refer to the same concepts. To avoid ambiguity, we unify the terminology concerning preferences. In this article, “agents” is used for individuals expressing their preferences, “alternatives” for items which are compared in preferences.

on their attribute and feature information (*e.g.* gender, location, personal interests, etc.). More recently, some works [14] discuss hybrid techniques using both node attributes and network topology for community detection. In this paper, we are particularly interested in attribute based techniques in the context of social networks. More precisely, we consider the case of interest-based social networks (*e.g.* Pinterest, Flickr, etc.) where agents sharing similar interests, opinions or viewpoints on some topics belong to the same community. In many real life applications, preferences (*i.e.* a preference describes how an agent orders any two alternatives) are considered to be very useful to efficiently express and model agent’s interests, needs or wishes. The aim of this work is to propose a novel community detection method based on clustering agents according to their preferences.

Few work has been done on clustering agents based on their preferences. Kamishima *et al.* in [10] proposed the k ’o-means clustering method, an adaptation of the k -means method, adjusted to support preference orders. In [17], the authors introduced a new community detection algorithm based on preference network. The communities are constructed according to the node preferences (*i.e.* each node gives information about its preferred nodes in order to be in the same group).

However, preferences are not always expressed firmly or consistently, sometimes a preference may be uncertain or imprecise facing an unknown situation, or conflicting when dealing with multiple sources. To the best of our knowledge, none of the work mentioned above has investigated preference-based clustering methods when preferences are imperfect (*i.e.* uncertain, imprecise or conflicting). To form meaningful groups of agents according to their preferences, a clustering algorithm need to capture the preference data structure and to cope with imperfect information.

In previous works [11, 19], a qualitative and expressive preference modeling strategy based on the theory of belief functions to model imperfect preferences was proposed. In this paper, starting from this model for agent’s preference modeling, we develop a preference-based clustering approach in the space of theory of belief functions. We discuss the clustering quality of our method by considering the Adjusted Rand Index (ARI) and silhouette coefficient as evaluation criteria. To highlight the relevance of the proposed solution, we perform experiments on synthetic and real data to compare our method with different preference modelings, reference in the field, and found the advantage in the expressiveness of the uncertainty and the conflict of the preferences.

Outline of the paper is as follows. In Sect. 2, we give background information related to preference orders, similarity measures over orders, and theory of belief functions. We then explain in Sect. 3 our previous preference model based on theory of belief functions and in Sect. 4 our clustering approach in detail. Experiments and their analysis are given in Sect. 5. We finish with the conclusion and perspectives in Sect. 6.

2 Basic Notions

2.1 Preference Order

Definition 1 (*Binary Relation*). Let $A = \{a_1, a_2, \dots, a_{|A|}\}$ be a finite set of alternatives, a **binary relation** O on the set A is a subset of the Cartesian product $A \times A$, that is, a set of ordered pairs (a_i, a_j) such that a_i and a_j are in A : $O \subseteq A \times A$ [13].

A binary relation satisfies any of the following properties: *reflexive, irreflexive, symmetric, antisymmetric, asymmetric, complete, strongly complete, transitive, negatively transitive, semitransitive, and Ferrers relation* [13]. The detailed definitions of the properties are not in the scope of this article.

Based on Definition 1, we denote the binary relation “prefer” by \succeq . The relation² $a_i \succeq a_j$ means “ a_i is at least as good as a_j ”. Inspired by a four-valued logic introduced in [1, 4, 11], we introduce four relations between alternatives. Given the alternative set A and a preference order \succeq defined on A , we have $\forall a_i, a_j \in A$, the four possible relations defined by:

- **Strict preference** denoted by P : $a_i \succ a_j$
 $(a_i \text{ is strictly preferred to } a_j) \Leftrightarrow a_i \succeq a_j \wedge \neg(a_j \succeq a_i)$
- **Inverse strict preference** denoted by $\neg P$: $a_j \succ a_i$
 $(a_i \text{ is inversely strictly preferred to } a_j) \Leftrightarrow \neg(a_j \succeq a_i) \wedge a_j \succeq a_i$
- **Indifference** denoted by I : $a_i \approx a_j$
 $(a_i \text{ is indifferent, or equally preferred, to } a_j) \Leftrightarrow a_i \succeq a_j \wedge a_j \succeq a_i$
- **Incomparability** denoted by J : $a_i \sim a_j$
 $(a_i \text{ is incomparable to } a_j) \Leftrightarrow \neg(a_i \succeq a_j) \wedge \neg(a_j \succeq a_i)$

A preferences structure $\langle P, I, J \rangle$ on multiple alternatives can therefore be presented by a binary relation [13].

Definition 2 (*Preference Structure*). A preference structure is a collection of binary relations defined on the set A such that for each pair a_i, a_j in A :

- at least one relation is satisfied
- if one relation is satisfied, another one cannot be satisfied.

The model for uncertain preferences detailed in [19] is compatible with quasi-orders while our dissimilarity measure is suitable for weak orders, which is a subset of quasi-orders, defined as [13]:

Definition 3 (*Weak Order*). Let O be a binary relation ($O = P \cup I$) on the set A , O being a characteristic relation of $\langle P, I \rangle$, the following three definitions are equivalent:

1. O is a weak order.
2. O is reflexive, strongly complete and transitive.
3. $\begin{cases} I \text{ is transitive} \\ P \text{ is transitive} \\ P \cup I \text{ is reflexive and complete.} \end{cases}$

² As $a_i \succeq a_j$ is equivalent to $a_j \preceq a_i$, to avoid repetitive comparisons between two alternatives, we assume $i > j$ in this article.

2.2 Dissimilarity Between Orders

To measure the dissimilarity between two preferences represented by total orders, metrics such as Euclidean distance and Kendall distance are often adopted. Given two preference orders O_1 and O_2 on the same alternatives, we give some basic concepts on such metrics.

Euclidean Distance. The rank function $r(O, a)$ denotes the position of the alternative a according to the order O . For example, for the order $O = a_1 \succ a_3 \succ a_2$, $r(O, a_1) = 1$ and $r(O, a_2) = 3$. Thus, for two orders O_1 and O_2 on the same alternative set A , Euclidean distance ($l^2 - norm$) between two orders is defined by:

$$d_{l^2}(O_1, O_2) = \sqrt{\sum_{a \in A} (r(O_1, a) - r(O_2, a))^2} \tag{1}$$

Kendall’s τ Distance and Fagin Distance. Kendall τ distance measures the dissimilarity with “penalty”. Fagin proposed a more general metric in [6] adapting for orders with indifference based on Kendall distance, we name it Fagin distance in this article. In Fagin distance, for alternatives a_i, a_j , the penalty between two orders O_1 and O_2 on a_i and a_j , denoted as $\bar{K}_{i,j}^{(p)}(O_1, O_2)$, is defined as follows:

- **Case 1:** a_i and a_j are in both O_1 and O_2 . If a_i and a_j are ordered in the same way (such as $a_i \succ a_j$ in both O_1, O_2), $\bar{K}_{i,j}^{(p)}(O_1, O_2) = 0$, this corresponds to “no penalty” for a_i and a_j . If a_i, a_j are ordered reversely (such as $a_i \succ a_j$ in O_1 while $a_i \prec a_j$ in O_2), the penalty of a_i, a_j $\bar{K}_{i,j}^{(p)}(O_1, O_2) = 1$.
- **Case 2:** a_i and a_j are tied in both O_1 and O_2 , $\bar{K}_{i,j}^{(p)}(O_1, O_2) = 0$. Intuitively, both partial orders agree that a_i and a_j are tied.
- **Case 3:** a_i and a_j are indifference in one of the partial order (say O_1) and of different rank in the other order (therefore O_2), we give a penalty parameter $\bar{K}_{i,j}^{(p)}(O_1, O_2) = p^3$.

Based on these cases, the *Kendall distance with penalty parameter p* (i.e. Fagin distance) is defined as follow:

$$K^{(p)}(O_1, O_2) = \sum_{i,j \in [1, |A|]} \bar{K}_{i,j}^{(p)}(O_1, O_2) \tag{2}$$

2.3 Belief Functions

The theory of belief functions (also referred to as Dempster-Shafer or Evidence Theory) was firstly introduced by Dempster [2] then developed by Shafer [16] as a general model of uncertainties. It is applied widely in information fusion

³ In our work, we take $p = 0.5$.

and decision making. By extending probabilistic and set-valued representations, it allows to represent degrees of belief and incomplete information in an unified framework. Let $\Omega = \{\omega_1, \dots, \omega_n\}$ be a finite set. A (*normalized*) *mass function* on Ω is a function $m : 2^\Omega \rightarrow [0, 1]$ such that:

$$m(\emptyset) = 0 \tag{3}$$

$$\sum_{X \subseteq \Omega} m(X) = 1 \tag{4}$$

The subsets X of Ω such that $m(X) > 0$ are called *focal elements* of m , while the finite set Ω is called *framework of discernment*. A mass function is called *simple support* if it has only two focal elements: $X \subseteq \Omega$ and Ω . A mass function having only one focal element $A \in \Omega$ is called a *categorical mass function*.

For example, if we consider the simple support mass $m(\omega_1 \cup \omega_2) = 0.8$, $m(\Omega) = 0.2$, this mass function represents an uncertainty with the degree 0.8 on the imprecise element $\{\omega_1 \text{ or } \omega_2\}$ and a partial ignorance with the degree 0.2 on Ω .

2.4 Distance on Belief Functions

Several distances can be used on belief functions and Jousselme distance is considered as a reliable similarity measure between different mass functions [5]. It considers coefficients on the elements composed by singletons. Jousselme distance is defined as follows, denoted by d_J :

Definition 4. *Let m_1 and m_2 be two mass functions on the same frame of discernment Ω , containing $|\Omega| = n$ mutually exclusive and exhaustive hypotheses. The distance between m_1 and m_2 is:*

$$d_J(m_1, m_2) = \sqrt{\frac{1}{2}(m_1 - m_2)^T \underline{D}(m_1 - m_2)} \tag{5}$$

where \underline{D} is the $2^n \times 2^n$ Jaccard matrix given by:

$$D(A, B) = \frac{|A \cap B|}{|A \cup B|}; A, B \subseteq \Omega. \tag{6}$$

3 Preference Model Under Uncertainty

In this section, we detail the preference model proposed in [19] as well as different dissimilarity measures for orders.

3.1 Problem Setting

We denote by U a set of agents, $U = \{u_1, u_2, \dots, u_{|U|}\}$, and A a set of mono-criterion alternatives, $A = \{a_1, a_2, \dots, a_{|A|}\}$. Every agent $u_i \in U$ expresses his/her preferences over A by a quasi order in the space of $A \times A$, denoted by O_i . Expert's preferences order O_i may come from two different sources $S = \{s_1, s_2\}$, denoted by O_1 and O_2 .

3.2 Preference Model on Belief Functions

The objective is to cluster the experts represented by their preferences under uncertainty. We consider the model from [19] to represent this uncertain preference by the theory of belief functions. The framework of discernment is defined on possible relations:

$$\Omega_{ij} = \{\omega_{ij}^1, \omega_{ij}^2, \omega_{ij}^3, \omega_{ij}^4\} \tag{7}$$

where $\omega_{ij}^1, \omega_{ij}^2, \omega_{ij}^3$ and ω_{ij}^4 , represent respectively $a_i \succ a_j, a_i \prec a_j, a_i \approx a_j$ and $a_i \sim a_j$. This procedure consists in two steps:

1. Initialization of mass functions
2. Clustering on quasi orders represented by mass functions.

This model is used in our contribution. Its utilization is described in detail in the following sections.

4 Contribution: Agent Clustering Based on Their Preferences

In this section, we explain how the agents are represented and clustered from two sources of preferences. The clustering procedure is straightforward, concisely illustrated in Fig. 1. The first block concerns the representation of agents and modeling of mass functions from two preference sources S_1, S_2 (Sects. 4.1 and 4.2). The second block concerns the measure of dissimilarity between agents (Sect. 4.3). The third block concerns clustering algorithm, we use EkNNclus algorithm in our work (Sect. 4.4).

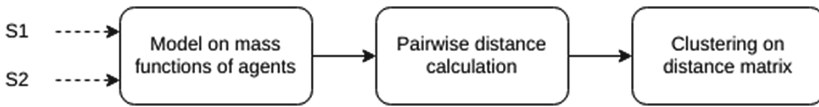


Fig. 1. Flowchart of preference based clustering

4.1 Representation of Agents

We consider the case that a group of $|U|$ agents expressing their preferences between each pair of alternatives from the set A . Therefore, the preferences of an agent u , denoted by O_u , is represented by a mass function on all possible alternative pairs:

$$O_u := [m_{1,2}; m_{1,3}; \dots; m_{1,K}; m_{2,3}; \dots; m_{|A|-1,|A|}] \tag{8}$$

Hence, for $|A|$ alternatives, the representation of an agent is made up by $\frac{(|A|-1)(|A|-2)}{2}$ mass functions.

4.2 Modeling of Mass Functions

In our model, we take advantage of the possibility of expressing on ignorance in the framework given by Eq. (7). Given two preference order sources O_1, O_2 from one agent, we interpret Fagin distance $d_F(O_1, O_2)$ as ignorance degree when conflict encountered. That is to say: for a_i, a_j ($i < j$) in both O_1, O_2 of agent u , with their ranking denoted by $r(O, a)$, the mass function value is given according to following conditions:

1. **Case 1:** a_i , and a_j are in the same relation in both O_1, O_2 (say $a_i \succ a_j$), $m_{i,j}$ is a categorical mass function ($m_{i,j}(\omega^1) = 1$).
2. **Case 2:** a_i and a_j are in the conflicting relations in O_1 and O_2 , respectively denoted as $\omega_{o_1}, \omega_{o_2} \in \Omega_{ij}, \omega_{o_1} \neq \omega_{o_2}$ (say $a_i \succ a_j$ in O_1 while $a_i \approx a_j$ in O_2 , $\omega_{o_1} = \omega^1, \omega_{o_2} = \omega^3$), the mass function values are given by:

$$\begin{aligned} m_{i,j}(\Omega) &= d_F(O_1, O_2) \\ m_{i,j}(\omega_{o_1}) &= m_{i,j}(\omega_{o_2}) = (1 - d_F(O_1, O_2))/2 \end{aligned} \quad (9)$$

4.3 Dissimilarity Between Different Agents

The dissimilarity measure is based on Jousselme distance [8] for mass functions. Given two mass functions modeling preference relations between alternatives i and j from agents u_1 and u_2 expressing preference orders O_1, O_2 . We denote Jousselme distance as $d_J(m_{ijO_1}, m_{ijO_2})$. The dissimilarity between two agents' preferences is denoted via Jousselme distance as:

$$d(O_1, O_2) = \sum_{j=1}^k \sum_{i=1, i < j}^k d_J(m_{ijO_1}, m_{ijO_2}) \quad (10)$$

where m_{ijO} denotes the mass function of alternative pair (a_i, a_j) according to the order O . Therefore, a normalized distance is given by

$$d_{Normalize}(O_1, O_2) = \frac{1}{Nb_{total}} d(O_1, O_2) \quad (11)$$

where $Nb_{total} = \frac{(|A|-1)(|A|-2)}{2}$, is the amount of all alternative pairs.

To simplify the expression, we use BF model to refer to our model and the corresponding dissimilarity function.

4.4 Unsupervised classifier–Ek-NN [3]

For dissimilarity spaces in which only pairwise distances are given (such as Kendall distance), the centroid of several agents is a metric k -center problem and is proved to be NP-hard. Therefore, we avoid using clustering methods requiring the calculation of centroid, such as k -means.

We applied Ek-NNclus method [3] as classifier. Ek-NNclus is a clustering algorithm based on the evidential k-nearest-neighbor rule, thus it requires only

the pairwise metric for k-nearest-neighbor searching. Another advantage of EkNNclus is that the number of clusters does not need to be determined in advance, only the neighborhood size k should be set. A determination of k is proposed in [20].

5 Experiments

Although the model was originally designed for preferences under uncertainty, we still wonder its quality for clustering on certain preferences. Thus the clustering quality of our model can be divided into two aspects: on certain preferences and on uncertain preferences.

5.1 Evaluation Criteria

With the similar aforementioned reasons, it's NP-hard to calculate centroids. Thus, we choose two evaluation criteria that do not require a cluster centroid calculation: Adjusted Rand Index (ARI) [7] for data with ground truth and silhouette coefficient [15] for any dataset.

We tested different metrics on synthetic certain and uncertain preferences. We also compared different metrics on a real world certain preferences from SUSHI data set [9].

In the following parts, we introduce the method of generating synthetic preferences and compare the clustering quality of different metrics. To simplify the experiments, all preferences are expressed in a space of 10 alternatives.

5.2 Certain Preferences

On Synthetic Data. Certain preferences are those who are from non-conflicting sources. In this case, we only consider and generate one source of preferences. To study the clustering quality, we firstly generate preferences with different ranges to their centroids. The data is generated in following steps in Algorithm 1.

By increasing the number of switching operations T , we obtain clusters with different densities.

To avoid random errors, we generate different preference sets 10 times and take the average value of ARI and silhouette score. Besides, the optimal parameter K in EkNN-clus algorithm varies with the size of data and distribution of the samples. The selection of K is not in the scope of this article. We test on various K and choose the one that returns the largest ARI and average silhouette coefficient⁴ as our result.

In Figs. 2, 3 and 4, ARI and silhouette coefficient performed on generated data with neighbors in different ranges (switch time from 1 to 3) and different sizes (neighbor size⁵ varies from 10 to 100) are illustrated.

⁴ Without special remark, we use term “silhouette coefficient” for “average” value on set of samples by default.

⁵ By saying neighbor size, we mean the number of samples in each cluster.

Algorithm 1. Generate preferences in $|C|$ clusters

Input: Cluster number $ C $ Switch time T neighbour size NS Alternative size in each order $ A $	8:	$dist_max = \sum_{i=1}^{i_c-1} d_{Kendall}(o_s, c_i)$
Output: $ C $ clusters of preferences <i>// Centroids initialization</i>	9:	centroid $c_{c_i} = o_s$
1: randomly generate centroid c_1 of $ A $ elements.	10:	end if
2: for i_c in $2 : C $ do	11:	end for
3: $dist_max = 0$	12:	end for <i>Generate neighbors</i>
4: for s in $1 : 5000$ do	13:	for each centroid o_c do
5: randomly generate preference o_s of $ A $ elements	14:	for ns in $1 : NS$ do
6: $dist_sum = \sum_{i=1}^{i_c-1} d_{Kendall}(o_s, c_i)$	15:	for t in $1 : T$ do
7: if $dist_sum > dist_max$ then	16:	randomly generate index i, j
	17:	exchange ranking order of a_i, a_j in o_c , making a new order
	18:	end for
	19:	end for
	20:	end for

According to these results, one can conclude that the BF model and Kendall distance have equivalent good quality both in terms of ARI and silhouette score, while Euclidean distance always has a poor quality. A high value in ARI usually corresponds to a high silhouette score, signifying a good clustering result.

On Real Data. SUSHI preference dataset [9] is collected from a survey on Japanese consumer preferences over different sushis. It has a data set containing 5000 complete strict rank orders (*i.e.* total orders) of 10 different kinds of sushi.

We applied these three metrics in clustering on real data of Sushi Preference Data Set. Figure 5 illustrates silhouette plots of clusters with different metrics.⁶ Kendall distance and BF model have similar quality. Euclidean distance has a relatively poor quality. This result is consistent with the synthetic data in Figs. 2, 3 and 4.

Among the three metrics, none of them has an absolutely high silhouette score (larger than 0.5). This is due to the quality of the data. SUSHI dataset does not guarantee the existence of obvious communities among the agents.

From both synthetic data and real world data without uncertainty, we observe that BF model and Kendall distance have very similar clustering. In fact, in case of certain problems, where all mass functions are categorical, for total orders, the normalized distance in BF model is degraded to Kendall distances. This can be easily proved by their definitions.

⁶ As different K in EKNN-clus algorithm returns different clustering results, we compare clustering result who returns relatively high silhouette coefficient.

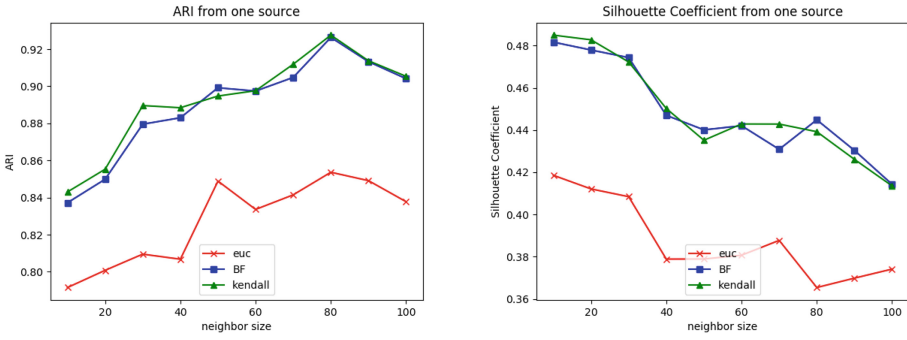


Fig. 2. ARI and silhouette coefficient, switch = 1

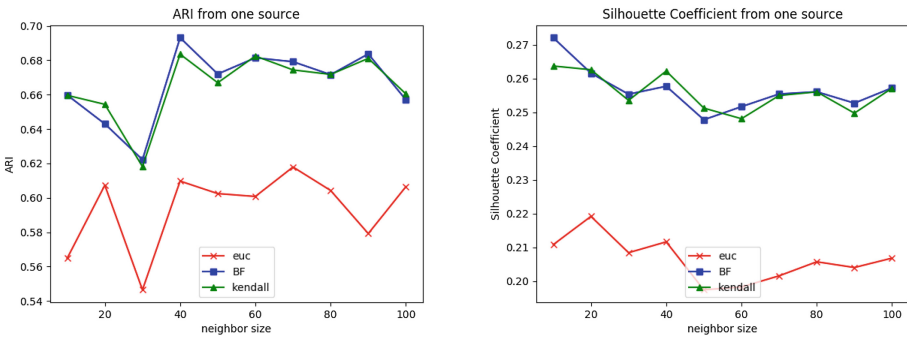


Fig. 3. ARI and silhouette coefficient, switch = 2

5.3 Uncertain Preferences

In this part, we suppose a case that two preferences are given with different representations: ranking and score. The ranking preferences are generated in the same way as in Subject. 5.2. Scores are generated by the following steps: scores range from 1 to 5 are generated respecting to a given rank preference. In this way, indifference relations are introduced, causing conflicts between two preference sources. Given a ranking preference O_r of 10 alternatives a_1 to a_{10} , the scores are generated by the following rules:

- For least preferred two alternatives (2 alternatives at the end of the O_r , *i.e.* ranking no. 9 and 10), we give score 1.
- For alternatives sorted at the positions 7 and 8, we give score 2.
- With the similar rule, for most preferred two sushis (ranking no. 1 and 2), we give score 5.

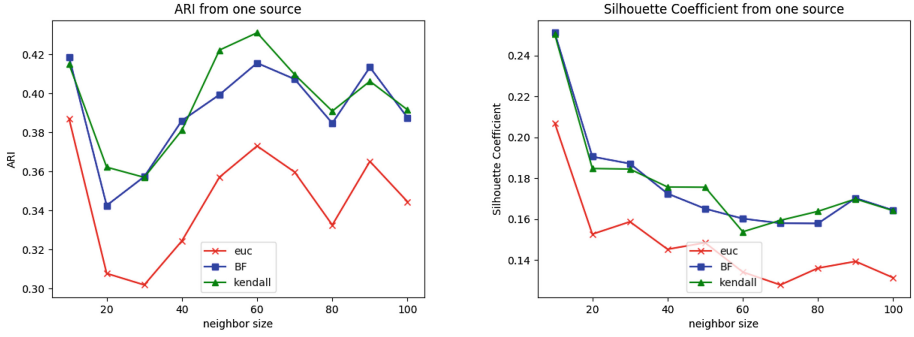


Fig. 4. ARI and silhouette coefficient, switch = 3

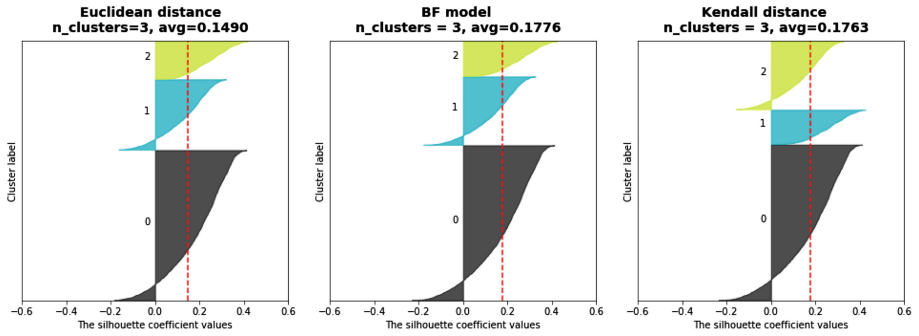


Fig. 5. Silhouette plots of different metrics on SUSHI

For example with:

$$a_1 \succ a_2 \succ a_3 \succ a_4 \succ a_5 \succ a_6 \succ a_7 \succ a_8 \succ a_9 \succ a_{10}$$

the scores are: $a_1 : 5, a_2 : 5, a_3 : 4, a_4 : 4, a_5 : 3, a_6 : 3, a_7 : 2, a_8 : 2, a_9 : 1, a_{10} : 1$.

Still, ARI and silhouette scores are applied as evaluation criteria. We compared our model with an average-based-euclidean metric calculated as follows:

Confronting a case of two preferences: ranking O_r and score O_s , the mean rank of alternative a_i is calculated:

$$\bar{r}(a_i) = \frac{1}{2}(r(O_r, a_i) + r(O_s, a_i))$$

Thus, agent u 's average preference order is represented by:

$$\bar{O}_u := [\bar{r}(a_1), \bar{r}(a_2), \dots, \bar{r}(a_{|A|})] \tag{12}$$

Therefore, the example above has a such vector:

$$[1, 1.5, 3, 3.5, 5, 5.5, 7, 7.5, 9, 9.5].$$

For Kendall distance, we calculate the distance matrix from rankings and scores, then take the average value as the combined distance. As indifference

relations exist in O_s , we apply Fagin distance for O_s . Given ranking preferences O_{r1}, O_{r2} and score preferences O_{s1}, O_{s2} , denoting the preference from agent u_1 and u_2 , the average distance is thus given by:

$$\bar{d}_{kendall}(u_1, u_2) = \frac{1}{2}(d_{kendall}(O_{r1}, O_{r2}) + d_{Fagin}(O_{s1}, O_{s2})) \quad (13)$$

We compared the model based on Euclidean distance Eq. (12), Kendall distance (13) and BF model given in Eq. (8).

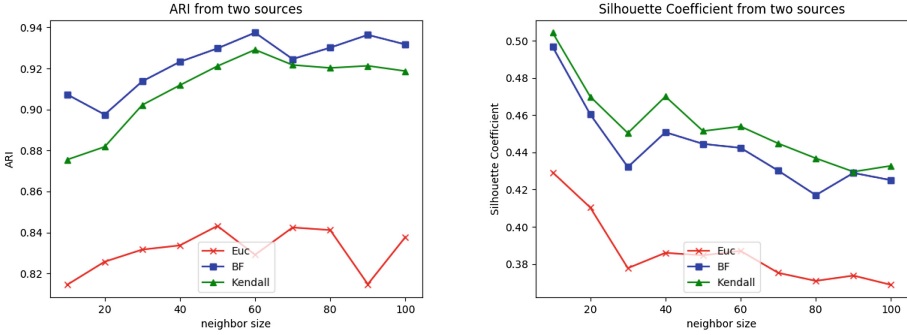


Fig. 6. ARI and silhouette coefficient on uncertain preferences, switch = 1

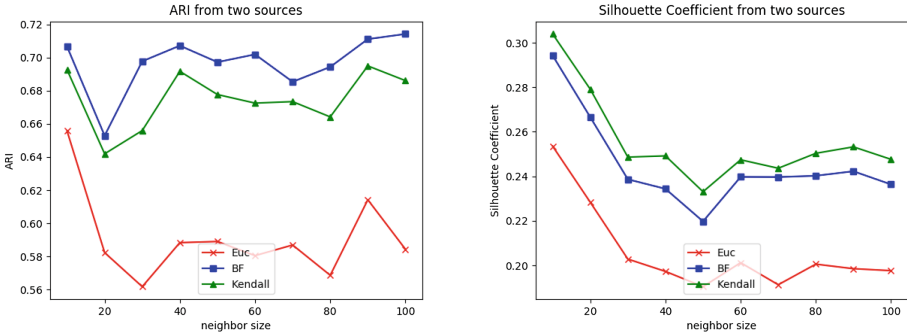


Fig. 7. ARI and silhouette coefficient on uncertain preferences, switch = 2

The results illustrated by these figures show the advantage of BF model over Euclidean distance and Kendall (Fagin) distance when dealing with two sources. Comparing Figs. 6, 7, and 8 from conflicting sources with Figs. 2, 3, and 4, we observe that both averaged Euclidean distance and Kendall distance are deteriorated more than BF model. The results prove the advantage of BF model on preferences under uncertainty. This advantage comes from the fact that in BF model, conflicts are partly interpreted as ignorance and have less impact in dissimilarity measuring. However, this compromise also causes a loss in criterion of silhouette coefficient.

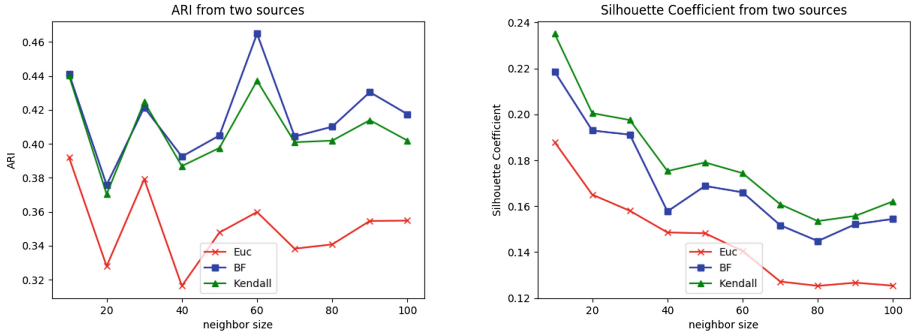


Fig. 8. ARI and silhouette coefficient on uncertain preferences, switch = 3

6 Conclusion and Perspectives

In this paper, we investigate the problem of clustering individuals according to their preferences, when dealing with multiple and conflicting sources (two in our case study). To cope with this issue, we apply the theory of belief functions (BF model) to express and interpret the contradictions and conflicts from different sources as uncertainty and ignorance. We introduce a new approach that captures the preference data structure and deal with uncertain information.

To highlight the relevance of the proposed solution, we perform experiments on synthetic and real data to compare our method with other preference models, and found the advantage in the expressiveness of the uncertainty and the incomparability of the preference orders. Indeed, we compare BF model on synthetic data between Euclidean distance and Kendall distance both in certain and uncertain cases, using Ek-NNclus algorithm for clustering. In certain cases, BF model has equivalent clustering-quality with Kendall distance and outperforms Euclidean distance. In uncertain cases, BF model has better clustering-quality over the other distances. We also applied this model on SUSHI preference data set and found that BF model has one of the most satisfying clustering-quality.

We applied the BF model on complete preference orders (*i.e. weak orders*) from only two sources. In the future, we will work on an ameliorated BF model dealing with several conflicting preference sources. In fact, the combination of preferences from multiple sources is a social choice problem, and different combination rules can be applied, corresponding to different complexity. Moreover, a more general dissimilarity measure method for incomplete orders (*i.e. quasi-orders*) is also in the scope of our future work.

References

1. Belnap, N.D.: A useful four-valued logic. In: Dunn, J.M., Epstein, G. (eds.) *Modern Uses of Multiple-valued Logic*. EPIS, vol. 2, pp. 5–37. Springer, Dordrecht (1977). https://doi.org/10.1007/978-94-010-1161-7_2
2. Dempster, A.P.: Upper and lower probabilities induced by a multivalued mapping. *Ann. Math. Statist.* **38**(2), 325–339 (1967)
3. Denoeux, T., Kanjanatarakul, O., Sriboonchitta, S.: EK-NNclus. *Know. Based Syst.* **88**(C), 57–69 (2015)
4. Elarbi, F., Bouadi, T., Martin, A., Ben Yaghlane, B.: Preference fusion for community detection in social networks. In: 24ème Conférence sur la Logique Floue et ses Applications. Poitiers, France, November 2015
5. Essaid, A., Martin, A., Smits, G., Ben Yaghlane, B.: A Distance-based decision in the credal level. In: *International Conference on Artificial Intelligence and Symbolic Computation (AISC2014)*, pp. 147–156. Sevilla, Spain, December 2014
6. Fagin, R., Kumar, R., Mahdian, M., Sivakumar, D., Vee, E.: Comparing and aggregating rankings with ties. In: *Proceedings of the Twenty-third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2004*, pp. 47–58. ACM, New York (2004)
7. Hubert, L., Arabie, P.: Comparing partitions. *J. Classif.* **2**(1), 193–218 (1985)
8. Jousselme, A.L., Maupin, P.: Distances in evidence theory: comprehensive survey and generalizations. *Int. J. Approx. Reason.* **53**(2), 118–145 (2012)
9. Kamishima, T.: Nantonac collaborative filtering: recommendation based on order responses. In: *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2003*, pp. 583–588. ACM, New York (2003)
10. Kamishima, T., Akaho, S.: Efficient clustering for orders. In: Zighed, D.A., Tsumoto, S., Ras, Z.W., Hacid, H. (eds.) *Mining Complex Data. Studies in Computational Intelligence*, pp. 261–279. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-540-88067-7_15
11. Masson, M.H., Destercke, S., Denoeux, T.: Modelling and predicting partial orders from pairwise belief functions. *Soft Comput.* **20**(3), 939–950 (2016)
12. Newman, M.E.J.: Modularity and community structure in networks. *Proc. Natl. Acad. Sci.* **103**(23), 8577–8582 (2006)
13. Öztürkçü, M., Tsoukiàs, A., Vincke, P.: Preference Modelling, pp. 27–59. Springer, New York (2005). <https://doi.org/10.1007/978-3-642-46550-5>
14. Qin, M., Jin, D., He, D., Gabrys, B., Musial, K.: Adaptive community detection incorporating topology and content in social networks. In: *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pp. 675–682. ACM (2017)
15. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* **20**, 53–65 (1987)
16. Shafer, G.: *A Mathematical Theory of Evidence*. Princeton University Press, Princeton (1976)
17. Tasgin, M., Bingol, H.O.: Community detection using preference networks. *Phys. A: Stat. Mech. Appl.* **495**, 126–136 (2018)
18. Yang, J., McAuley, J., Leskovec, J.: Community detection in networks with node attributes. In: *13th International Conference on Data Mining (ICDM 2013)*, pp. 1151–1156. IEEE (2013)

19. Zhang, Y., Bouadi, T., Martin, A.: Preference fusion and Condorcet's paradox under uncertainty. In: 20th International Conference on Information Fusion, FUSION 2017, pp. 1–8. Xi'an, China (2017)
20. Zhang, Y., Bouadi, T., Martin, A.: An empirical study to determine the optimal k in EK-NNclus method. In: 5th International Conference on Belief Functions, BELIEF 2018. Compiègne, France (2018)



A Novel Committee–Based Clustering Method

Sonia Fiol-Gonzalez¹, Cassio Almeida^{1,2}, Simone Barbosa¹,
and H elio Lopes¹(✉)

¹ Departamento de Inform tica, Pontif cia Universidade Cat lica do Rio de Janeiro,
Rio de Janeiro, Brazil

{sgonzalez, calmeida, simone, lopes}@inf.puc-rio.br

² ENCE - Instituto Brasileiro de Geografia e Estat stica, Rio de Janeiro, Brazil

Abstract. It is well recognized that clustering algorithms play an important role in data analysis. For a successful application of these algorithms, it is crucial to determine the relevant features in the original dataset. To deal with this problem there are efficient techniques for feature selection in the literature. Moreover, it is also well known that, in the clustering task, it is also difficult to define an adequate number of clusters. This paper proposes a new ensemble clustering method that is comprised of three stages: the first generates the clustering ensemble, the second combines the results of the multiple clustering scenarios generated, and the last one creates a new partition using the combined data. To generate the clustering ensemble, the method combines feature selection strategies and clustering with various numbers of clusters to produce a similarity matrix. This similarity matrix is then used to compute the final clustering output. Experiments performed using seven well known datasets showed the effectiveness of the proposed technique.

Keywords: Feature selection · Clustering methods
Similarity matrix · Ensemble methods · Unsupervised learning

1 Introduction

Nowadays, biomedicine, bioinformatics, sociology, economy, marketing, pattern recognition, and computer vision are examples of research areas that use machine learning to generalize behaviors. In this context, machine learning techniques come to reveal the natural structure of the data in those areas. A specific kind of problem within machine learning is to group the elements of a dataset according to their similarities, so as to define classes of objects with certain common properties and behavior. This is typically called a clustering problem. Because there is no known association of a value (discrete/continuous) to each element in the dataset, i.e., the class to which each object belongs is unknown, the clustering problem is classified as an unsupervised learning problem. In particular,

Supported by CAPES and CNPq.

clustering is challenging due to the lack of generalized algorithms. Instead, there are specific domain solutions using particular key details according to the application area [25].

Datasets can have large number of features, but usually not all features are relevant for clustering, so it is necessary to select a subset of them. Feature selection has thus become an important tool for decision support systems that use thousands of features [5]. The importance of this technique lies not only in the decrease in execution time, but also in the improvement of the clustering quality. There are two major kinds of feature selection techniques: attribute selection [3, 13] and attribute transformation [7]. These techniques are being widely used to remove related or irrelevant attributes in machine learning algorithms, both in supervised learning [2], and in unsupervised learning [9].

In addition to using feature selection techniques, the scientific machine learning community has increasingly made use of committees [27]. Combining committees with various machine learning techniques has proven to be effective and versatile for solving real life problems [19].

The objective of this work is to improve clustering results by combining multiple clustering and features selection algorithms. To do so, we propose a new ensemble clustering method that is comprised of three stages. The first generates the clustering ensemble by combining multiple partitions of the dataset using traditional clustering methods. To construct this ensemble, we combine feature selection algorithms and vary the number of clusters. The second stage uses the results of the multiple scenarios generated in the first stage to create a distance matrix between the elements. To do so, we propose a weighting function based on the silhouette coefficient to represent the strength of the connection of two different objects of the data. This weight is computed for each scenario in the ensemble. The last stage creates a new partition using this matrix. Experiments performed using seven well known datasets and a real-world dataset showed the effectiveness of the proposed method. In addition, this work explores the resulting matrix not only as a distance matrix for clustering generation [11], but also as an adjacency matrix with weights where statistical network analysis [15] can be applied.

This paper is organized as follows. Section 2 presents the background and related work. Section 3 explains in details the proposed method. Section 4 presents the experiments and the results in seven known datasets. Finally, Sect. 5 presents the conclusions and suggests future work.

2 Related Work

This section presents a literature review with some feature selection algorithms, as well as ensemble methods used in unsupervised learning.

Feature Selection Algorithms: In machine learning and in statistics, feature selection is a strategy for selecting a subset of relevant features to build robust learning models [21]. Selecting the most relevant feature subset based on certain evaluation criteria is essentially a combinatorial optimization problem, which is

computationally expensive [4]. Feature selection methods are largely studied separately according to the type of learning: supervised or unsupervised [28]. Two possible methods for unsupervised feature selection in this context are Sequential Forward Selection (SFS) and Sequential Backward Elimination or Selection (SBS).

The algorithm for SFS [8] begins by adding, to the empty set of best features, the individual feature which obtains the best performance. Next, all the remaining features are tested together with this single feature to see which combination performs the best. The pair of features providing the best result is assigned to the best set of features. Continue adding features to the set in this way until the result of the evaluation function reaches some pre-determined threshold value. The search stops at this point. The optimal set of features returned by this algorithm are those features contained in the best set.

The SBS algorithm works in the opposite manner when compared to SFS, since it removes features instead of adding them. Starting with a set of all features, at each step in the algorithm, the feature whose removal results in the smallest decrease of the value of the evaluation function is removed. In some cases, removing a feature may actually increase the value of the evaluation function, and so it should be removed as well. The algorithm halts when it is impossible to remove any single remaining feature without crossing a predefined threshold [8].

The disadvantage in SFS is called “nesting effect” since, once selected, features cannot be later discarded. Likewise, in the case of SBS, once discarded, a feature cannot be re-selected [20]. Both selection procedures use search techniques which avoid exhaustive enumeration [12]. Even though the selection of the optimal subset is not guaranteed in any of these techniques, we decided to use them because they are both simple to implement and very efficient.

Ensemble Clustering: Different clustering methods applied to the same dataset produce different results. Moreover, each method has both advantages and disadvantages which depend on the data set. There is no silver bullet. So, the idea of combining multiple clustering results seems reasonable. The clustering ensemble technique aims to combine multiple clustering results from the same dataset into a final clustering. Several papers make references to this solution [10, 11, 14, 22–26]. They all agree that it is a difficult problem.

There are several approaches to generate cluster ensembles, but in general they follow one of two main directions. The first one is to use different data representations and structures, such as graphs, vectors or strings, as well as to reduce the dimensionality of the problem by selecting some of the features to create different feature spaces. The second approach is to use different clustering algorithms and parameters, i.e., to apply multiple methods, the same method with different parameters, or even to use distinct dissimilarity measures to generate the desired partition [11].

The problem of combining the partitions into the final clusters has been addressed by several works. Among the most popular clustering ensemble techniques we find methods based in Co-association Matrix, Graph and Hypergraph

Partitioning, and Finite Mixture Models. There are other techniques available in [24]. Strehl et al. [22] presented three effective and efficient techniques based on hypergraph representation for obtaining high quality combiners. The first one is Cluster-based Similarity Partitioning Algorithm (CSPA); this technique uses the relation between the elements to build a co-association matrix and then reclusters the objects. The second one is HyperGraph Partitioning Algorithm (HGPA); this technique build clusters by removing edges from the hypergraph. The third one is Meta-Clustering Algorithm (MCLA). This method is based on the analysis of the similarity between clusters, treating them as nodes in the hypergraph and finally creating a meta-cluster using this hypergraph. Huang et al. [14] presented an algorithm based on bipartite graphs called Graph Partitioning with Multi-Granularity Link Analysis (GP-MGLA). They define levels of granularity – instances, ensemble, and final clustering –, then build a bipartite graph using instances and clusters to extract the consensus clustering. Another approach based on the finite mixture model was provided in [23] as a probabilistic model using the Expectation Maximization (EM) algorithm [6] in order to assign labels to elements in the partitions. In addition, [11] explored the evidence accumulation (EAC) matrix and presented a framework based on the co-association technique for extracting a consensus clustering from the clustering ensemble, applying Average Link (EAC-AL) and Single Link (EAC-SL). Based on this work, Wang et al. [25] proposed to incorporate probability theory into the evidence accumulation matrix to create a probability accumulation. Yi et al. [26] proposed a solution to overcome uncertainty in the data by examining the different clusterings in the ensemble and selecting the ones that agreed the most, called Ensemble Clustering Matrix Completion (ECMC). Another approach to the EAC presented in [14] was the Weighted Evidence Accumulation Clustering (WEAC) method. This method include weights to penalize low quality clusterings and agglomerative methods (WEAC+SL, WEAC+AL, WEAC+CL) to obtain the consensus partition.

3 The 3-Stage Ensemble Clustering Method

Clustering ensemble has become a modern approach and a cutting edge technique when solving clustering problems with unconventional structures, due to its learning capabilities. It is a flexible technique because, according to the number of partitions, co-relation criteria and the number of iterations can be adjusted to solve a range of widely diverse problems. Even when the focus of the ensemble methods is to generate a consensus result which best fits the natural structure of the dataset, they all try different approaches and present diverse techniques to solve the problem. In general, there is no single way of dealing with ensemble clustering but as a method with fixed and variable spots to fill in.

Inspired on the similarity matrix [26], co-association or evidence accumulation matrix [11], probability accumulation matrix [25] and using the silhouette coefficient as in [16] to measure the association between two elements, this work proposes a new ensemble clustering method to create a similarity matrix. This

new method comprises three stages: Stage 1 generates the clustering ensemble, Stage 2 combines the results of the multiple scenarios generated, and Stage 3 creates a new partition using the combined data.

The two important characteristics of the method concern the resulting relevant features and the number of times two objects are assigned to the same cluster. Our method is independent of the clustering method and feature selection algorithms. The feature selection methods currently used are the sequential SFS and SBS. For each feature subset selected, the method uses various clustering methods, currently K-Means, K-Medoids (PAM) and Hierarchical Clustering with Average Link.

In Stage 1, the ensemble is generated by combining the feature selection methods, varying the clustering methods and the number of clusters k , where k varies from 2 to an input parameter l (by default, $l = N/2$ where N is the total number of elements). Each possible combination is called a scenario. A solution belongs to a single scenario and can be defined as a tuple $\langle k, clm, fsm, rf, silh, fm \rangle$, where: k is the number of clusters; clm corresponds to the clustering method used; fsm is the method of feature selection employed; rf are the relevant features in the resulting clustering; $silh$ is the value of the silhouette coefficient and fm corresponds to the resulting frequency matrix. The frequency matrix (fm) contains in the position (i, j) the median silhouette coefficient of the elements i and j , normalized between 0 and 1 (see Algorithm 1). The average (mean) silhouette coefficient can be affected by outliers in the cluster, so we chose to use the median to reduce the effect of outliers on the measure.

In Stage 2, the similarity matrix sim is generated from the multiple scenarios, as shown in Algorithm 2. To quantify the strength of the bind between two objects, the frequency matrices are filled and finally divided by the number of scenarios, creating the similarity matrix sim as shown in Algorithm 2, allowing further exploration of objects and their relationships. The bind between two objects i and j is defined as the element $sim_{i,j}$ and its value lies within $[0, 1]$. It can be interpreted as a probability of element i being in the same cluster as element j . The relation between elements i and j is considered strong if $sim_{i,j} > 0.50$. This means that they are in the same group in over half of the clusterings. Conversely, the relation is considered weak when $sim_{i,j} < \alpha$.

In the result of a classical clustering algorithm, any two objects can either be in the same cluster or in different clusters, regardless of their bind. In other words, an element can have a strong bind with an element outside its cluster and a weak bind with elements inside it. This can happen, for instance, when an element is in the boundaries between a group of clusters, so it can be associated with any one of them, or when the clustering method has a random component like K-Means.

The main objective of our method is to obtain the similarity matrix sim and use it to verify the strength of the binds. As part of Stage 3, each element $sim_{i,j}$ of this matrix can be transformed using $dm_{i,j} = 1 - sim_{i,j}$ to define a distance matrix dm that could be used in other clustering processes.

Algorithm 1. Algorithm to Create a Frequency Matrix

```

procedure CREATEFREQUENCYMATRIX(clusteringMatrix, silhouettesMatrix)
  fm ← initialized with 0
  n ← number of rows of clusteringMatrix
  m ← number of columns of clusteringMatrix
  for d = 1 to m do
    s ← silhouettesMatrixd
    for each i, j = (1, 1) to (n, n) do
       $sm_{i,j} = \frac{s_i + s_j + 2}{4}$ 
      if clusteringMatrixd[i] == clusteringMatrixd[j] then
        fmi,j ← fmi,j + smi,j
  return  $\frac{fm}{m}$ 

```

Algorithm 2. Algorithm to Generate and Combine the Multiple Scenarios

```

procedure GENERATESCENARIOS(data, K, CMethods, FSMethods, threshold)
  sol_scen ← list()
  for each k in K do
    for each cm in CMethods do
      for each fsm in FSMethods do
        [rf, clMatrix, sMatrix, silh] ← CreateClustering(data, k, cm, fsm,
threshold)
        fm ← CreateFrequencyMatrix(clMatrix, sMatrix)
        scenario ← [k, clm, fsm, rf, silh, fm]
        add scenario to sol_scen
  sum_fm ← add the fm from multiple scenarios
  return sim ← sum_fm/|sol_scen|

```

Varying the number of clusters and the clustering methods, a new set of partitions is generated using the dissimilarity matrix dm as distances between the elements. Notice that here the features are not used to calculate distances like in the first stage of the method, only the relationships between the elements. Finally, the partition with the best silhouette coefficient is used to generate a recommendation of the final clustering and the number of clusters.

Having this in mind, the final recommendation is represented as a tuple $\langle sim, rfs, fcl, k \rangle$ where sim is the similarity matrix, rfs is the set of relevant features, fcl is the final clustering and k is the number of clusters in fcl . This recommendation is made given the initial parameters, namely: a set of cluster numbers, a set of clustering methods, and a set of feature selection methods.

4 Experiments and Results

This section presents the experiments and the results obtained. To evaluate the performance of the method, it first presents a comparison between several clustering methods like K-Means, PAM, HC with Average Link and the

proposed method, and then it shows another comparison between clustering methods based on ensemble techniques and our method. The comparison measures the quality of the resulting clusters using normalized mutual information.

In this experiment were used seven well known datasets, which are used for comparison in [26] and in [14]. All of them are available in the UCI Machine Learning Repository [17]. Table 1 provides an overview of these datasets.

Table 1. Overview of datasets.

	Dataset	Nb. Instances	Nb. Attributes	Nb. Classes
1	Iris	150	4	3
2	Wine	178	13	3
3	Seeds	210	7	3
5	Glass	214	9	6
4	Breast Cancer	683	10	2
6	Yeast	1484	8	10
7	Segmentation	2310	19	7

In order to compare the clustering methods, several quality measures were adopted: Proportion of correct classifications (P), Adjusted Rand index (AR), Variation in Information (VI) [18], Silhouette coefficient (Sil) and Normalized Mutual Information (NMI).

The clustering methods used are available in R [1]: K-Means and HC-AL from the `stats` package, PAM from `cluster` package; the measures Silhouette from `cluster` package, P, AR and VI from `MixSim` package and, finally, NMI from the `infotheo` package.

The experiments were divided in two groups: individual clustering algorithms and ensemble clustering. Our methods will be represented in this section by the labels Meth - PAM and Meth - HC-AL. The first is an implementation of our ensemble clustering algorithm using PAM as the clustering method, and for the second we use the Hierarchical clustering algorithm with average link. In all these experiments we consider $\alpha = 20\%$.

The individual clustering methods used in this experiment are K-Means, PAM and HC with Average Link. Table 2 shows the results. We highlighted in boldface the best results of each method according to Proportion of correct classifications (P) and Normalized Mutual Information (NMI).

In the first and the second datasets, our method correctly classifies almost all the elements, achieving much better results than the individual methods. In the other datasets the results are very similar. In the Variation in Information criterion, we can see that the method usually has lower values when compared with the classical clustering algorithms.

The normalized mutual information is better in the first two datasets and similar in the remaining ones. Notice that the silhouette values for our methods

Table 2. Performance of the K-Means, PAM and HC-AL clustering methods and the proposed method, with corresponding quality measures.

	Dataset	Method	P	AR	VI	Sil	NMI
1	Iris	K-Means	0.580	0.433	0.818	0.550	0.593
2		PAM	0.847	0.642	0.713	0.406	0.675
3		HC-AL	0.687	0.562	0.524	0.696	0.728
4		Meth - PAM	0.960	0.886	0.298	0.362	0.864
5		Meth - HC-AL	0.960	0.886	0.298	0.362	0.864
6	Wine	K-Means	0.966	0.897	0.270	0.360	0.876
7		PAM	0.910	0.741	0.471	0.311	0.783
8		HC-AL	0.388	-0.005	1.184	0.178	0.031
9		Meth - PAM	0.978	0.933	0.192	0.371	0.912
10		Meth - HC-AL	0.983	0.949	0.156	0.370	0.928
11	Seeds	K-Means	0.919	0.773	0.598	0.453	0.728
12		PAM	0.910	0.747	0.626	0.468	0.714
13		HC-AL	0.881	0.686	0.770	0.494	0.649
14		Meth - PAM	0.867	0.653	0.796	0.407	0.637
15		Meth - HC-AL	0.881	0.690	0.681	0.401	0.689
16	Glass	K-Means	0.453	0.170	2.021	0.369	0.315
17		PAM	0.430	0.157	2.033	0.396	0.306
18		HC-AL	0.379	0.019	1.618	0.412	0.149
19		Meth - PAM	0.439	0.197	1.997	0.251	0.348
20		Meth - HC-AL	0.472	0.218	1.521	0.244	0.354
21	Breast Cancer	K-Means	0.958	0.836	0.343	0.511	0.733
22		PAM	0.965	0.863	0.299	0.509	0.768
23		HC-AL	0.647	-0.003	0.665	0.626	0.011
24		Meth - PAM	0.956	0.830	0.343	0.508	0.732
25		Meth - HC-AL	0.947	0.798	0.382	0.509	0.699
26	Yeast	K-Means	0.414	0.163	2.704	0.154	0.289
27		PAM	0.380	0.156	2.747	0.182	0.289
28		HC-AL	0.324	0.019	1.813	0.533	0.120
29		Meth - PAM	0.291	0.095	3.078	0.100	0.207
30		Meth - HC-AL	0.312	-0.021	2.459	0.215	0.104
31	Segmentation	K-Means	0.532	0.413	1.625	0.258	0.575
32		PAM	0.677	0.507	1.528	0.271	0.606
33		HC-AL	0.435	0.238	1.192	0.361	0.624
34		Meth - PAM	0.646	0.507	1.349	0.282	0.646
35		Meth - HC-AL	0.578	0.442	1.115	0.474	0.675

Table 3. NMI values indicating the performances of different ensemble clustering methods and the proposed method.

	Method	Breast_Cancer	Iris	Seeds	Yeast	Wine	Segmentation
1	GP_MGLA	0.618	0.695	0.514	0.167	0.717	0.549
2	WEAC_AL	0.596	0.673	0.517	0.147	0.664	0.533
3	WEAC_CL	0.073	0.653	0.197	0.093	0.177	0.317
4	WEAC_SL	0.030	0.493	0.317	0.046	0.235	0.420
5	HBCF	0.648	0.640	0.493	0.181	0.647	0.491
6	WCC	0.459	0.539	0.493	0.208	0.581	0.527
7	EAC_AL	0.512	0.667	0.399	0.109	0.444	0.503
8	EAC_CL	0.058	0.497	0.206	0.065	0.168	0.323
9	EAC_SL	0.010	0.632	0.244	0.034	0.104	0.413
10	ECMC_AL	0.399	0.140	0.126	0.021	0.154	0.081
11	ECMC_CL	0.358	0.181	0.146	0.022	0.159	0.102
12	ECMC_SL	0.259	0.272	0.064	0.017	0.078	0.026
13	SRS_AL	0.519	0.676	0.438	0.122	0.254	0.513
14	SRS_CL	0.489	0.648	0.356	0.116	0.407	0.530
15	SRS_SL	0.029	0.661	0.344	0.034	0.001	0.411
16	WCT_AL	0.075	0.673	0.403	0.120	0.478	0.494
17	WCT_CL	0.110	0.644	0.326	0.095	0.396	0.492
18	WCT_SL	0.124	0.650	0.289	0.035	0.184	0.416
19	ECMC	0.519	–	–	0.277	–	0.540
20	MCLA	0.501	–	–	0.113	–	0.024
21	CSPA	0.489	–	–	0.161	–	0.502
22	HPGA	0.382	–	–	0.106	–	0.387
23	EAC_SL	0.442	–	–	0.092	–	0.415
24	EAC_AL	0.418	–	–	0.271	–	0.530
25	QMI	0.510	–	–	0.202	–	0.537
26	DiCLENs	0.512	–	–	0.089	–	0.349
27	Meth - PAM	0.732	0.864	0.637	0.207	0.912	0.646
28	Meth - HC-AL	0.699	0.864	0.689	0.104	0.928	0.675

are usually worse than for the individual methods. This shows that high values of the silhouette coefficient do not imply a clustering fitting the natural data structure, *i.e.*, it does not reflect the correct classifications as represented by P .

In order to compare the results obtained with the method, we used several algorithms, namely MCLA, CSPA, HPGA, EAC, GP-MGLA, WEAC and ECMC. In addition, we used six other methods presented in [14, 26]. Table 3 shows these results. We highlighted in boldface the two best results of each

method using the NMI measure. In Table 3, the comparisons are only taking into account the NMI coefficient. The proposed method has a better result in five of the six datasets. In the other one the best results of our method is 0.207, relative close to ECMC with 0.277. Notice here that in the Wine dataset the best result of the methods is 0.717 and the worst result in the proposed method is 0.912. Notice that the ensemble clusters methods from rows 19 to 26 (results taken from [26]) are not better than a simple K-Means (see the NIM column of Table 2), whereas the results of our methods are better or similar.

5 Conclusion and Future Work

This work presented a new ensemble clustering method, which creates a similarity matrix combining multiple partitions of the dataset using feature selection and clustering methods. Finally, it joined all partitions in order to obtain a consensus one that uses a transformation of the previously computed similarity matrix as an input to other clustering process. The experiments show evidence of good performance of our method: it is better or similar when compared with the individual clustering methods. Moreover, when it is compared with the ensemble clustering methods, it also has either better or competitive results. We plan to scale this algorithm to deal with large database by using a suitable sampling technique in hierarchical scheme.

References

1. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2015). <https://www.R-project.org/>
2. Armanfard, N., Reilly, J.P., Komeili, M.: Local feature selection for data classification. *IEEE Trans. Pattern Anal. Mach. Intell.* **38**(6), 1217–1227 (2016)
3. Bolón-Canedo, V., Sánchez-Marroño, N., Alonso-Betanzos, A.: A review of feature selection methods on synthetic data. *Knowl. Inf. Syst.* **34**(3), 483–519 (2013)
4. Cai, D., Zhang, C., He, X.: Unsupervised feature selection for multi-cluster data. In: *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 333–342. ACM (2010)
5. Chandrashekar, G., Sahin, F.: A survey on feature selection methods. *Comput. Electr. Eng.* **40**(1), 16–28 (2014)
6. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. *J. R. Stat. Soc. Ser. B (Methodological)* **39**(1), 1–38 (1977)
7. Dhir, C.S., Lee, J., Lee, S.Y.: Extraction of independent discriminant features for data with asymmetric distribution. *Knowl. Inf. Syst.* **30**(2), 359–375 (2012)
8. Doak, J.: An evaluation of feature selection methods and their application to computer security. University of California, Computer Science (1992)
9. Farahat, A.K., Ghodsi, A., Kamel, M.S.: Efficient greedy feature selection for unsupervised learning. *Knowl. Inf. Syst.* **35**(2), 285–310 (2013)
10. Fern, X.Z., Brodley, C.E.: Random projection for high dimensional data clustering: a cluster ensemble approach. *ICML* **3**, 186–193 (2003)

11. Fred, A.L., Jain, A.K.: Combining multiple clusterings using evidence accumulation. *IEEE Trans. Pattern Anal. Mach. Intell.* **27**(6), 835–850 (2005)
12. Fukunaga, K.: *Introduction to Statistical Pattern Recognition*, 2nd edn. Academic Press Professional Inc., San Diego (1990)
13. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. *J. Mach. Learn. Res.* **3**, 1157–1182 (2003)
14. Huang, D., Lai, J.H., Wang, C.D.: Combining multiple clusterings via crowd agreement estimation and multi-granularity link analysis. *Neurocomputing* **170**, 240–250 (2015)
15. Kolaczyk, E.D., Csárdi, G.: *Statistical Analysis of Network Data with R*. Springer, New York (2014). <https://doi.org/10.1007/978-1-4939-0983-4>
16. Li, N., Latecki, L.J.: Clustering aggregation as maximum-weight independent set. In: *Advances in Neural Information Processing Systems*, pp. 782–790 (2012)
17. Lichman, M.: UCI machine learning repository (2013). <http://archive.ics.uci.edu/ml>
18. Melnykov, V., Chen, W.C., Maitra, R.: MixSim: an R package for simulating data to study performance of clustering algorithms. *J. Stat. Softw.* **51**(12), 1–25 (2012)
19. Mendes-Moreira, J., Soares, C., Jorge, A.M., Sousa, J.F.D.: Ensemble approaches for regression: a survey. *ACM Comput. Surv. (CSUR)* **45**(1), 10 (2012)
20. Murty, M.N., Devi, V.S.: *Pattern Recognition: An Algorithmic Approach*. Springer, London (2011). <https://doi.org/10.1007/978-0-85729-495-1>
21. Peter, T.J., Somasundaram, K.: Study and development of novel feature selection framework for heart disease prediction. *Int. J. Sci. Res. Publ.* **2**(10), 1–7 (2012)
22. Strehl, A., Ghosh, J.: Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.* **3**, 583–617 (2002)
23. Topchy, A.P., Jain, A.K., Punch, W.F.: A mixture model for clustering ensembles. In: *SDM*, pp. 379–390. SIAM (2004)
24. Vega-Pons, S., Ruiz-Shulcloper, J.: A survey of clustering ensemble algorithms. *Int. J. Pattern Recognit. Artif. Intell.* **25**(03), 337–372 (2011)
25. Wang, X., Yang, C., Zhou, J.: Clustering aggregation by probability accumulation. *Pattern Recognit.* **42**(5), 668–675 (2009)
26. Yi, J., Yang, T., Jin, R., Jain, A.K., Mahdavi, M.: Robust ensemble clustering by matrix completion. In: *IEEE 12th International Conference on Data Mining*, pp. 1176–1181. IEEE (2012)
27. Zhang, C., Ma, Y.: *Ensemble Machine Learning*. Springer, New York (2012). <https://doi.org/10.1007/978-1-4419-9326-7>
28. Zhao, Z., Liu, H.: Spectral feature selection for supervised and unsupervised learning. In: *Proceedings of the 24th International Conference on Machine Learning*, pp. 1151–1157. ACM (2007)



KMN - Removing Noise from K-Means Clustering Results

Benjamin Schelling^{1(✉)} and Claudia Plant^{1,2}

¹ University of Vienna, Vienna, Austria
benjamin.schelling@univie.ac.at

² ds:UniVie, Vienna, Austria

Abstract. K-Means is one of the most important data mining techniques for scientists who want to analyze their data. But K-Means has the disadvantage that it is unable to handle noise data points. This paper proposes a technique that can be applied to the k-means Clustering result to exclude noise data points. We refer to it as KMN (short for K-Means with Noise). This technique is compatible with the different strategies to initialize k-means and determine the number of clusters. Moreover, it is completely parameter-free. The technique has been tested on artificial and real data sets to demonstrate its performance in comparison with other noise-excluding techniques for k-means.

1 Introduction

A scientist who has not had much contact with data mining will use the simplest algorithms when he decides to use data mining techniques. The simplest and best known is probably k-means [10]. More refined techniques, that offer the possibility of achieving better results are likely to be applied at a later stage, once the scientist has become familiar with the automatic labelling of the data and has learned to appreciate the additional information that such techniques might yield. K-means is something like a gateway to data mining techniques. It has many advantages that predestine it for this purpose: its simplicity, the comparatively good results and its runtime. But k-means also has some disadvantages that are not to be neglected: The initialization that determines which (local) optimum the algorithm converges to, the need to set the k parameter and its inability to handle noise. This can be seen in Fig. 1. K-means will add each data point to a cluster, since the possibility, that a data point is a noise point is simply not supported in the algorithm. The first two problems, initialization and setting k , were examined in detail and various (sometimes very capable) strategies have been proposed, the last problem however received less recognition. This may be partly due to the fact that publicly available data sets rarely contain noise, which is why k-means-based techniques that find and characterize noise did not seem so important at first. Recently, however, there have been more and more techniques that take noise into account when clustering.

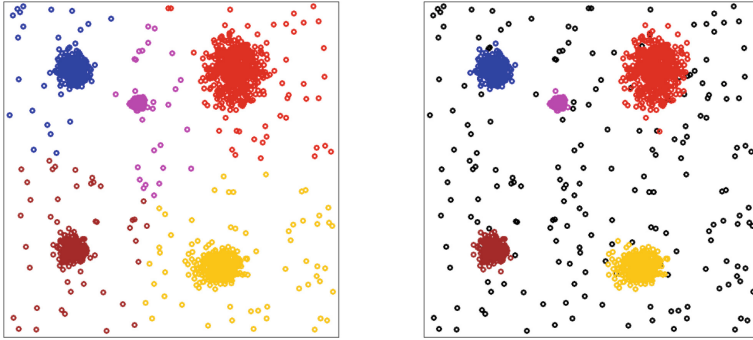


Fig. 1. A typical clustering result of k-means, if the value of k is chosen correctly, and how the clustering actually should look like.

1.1 Related Work

Clustering in the presence of noise is a classic field of research in data mining. Many techniques like DBSCAN [6] have been suggested and studied extensively. However, few of them are based on k-means, hence will not be a likely first choice for a new user of data mining techniques. The focus here is exclusively on techniques which adapt k-means to noisy data sets and thus offer the possibility to stay within the framework of k-means while they are still dealing with noisy data sets.

The best known of these clustering-in-the-presence-of-noise techniques based on k-means is k-means-- from Chawla et al. [5]. The problem is that it asks for the number of outliers as a parameter for the functionality of the technique. Chawla et al. stated in their paper that “all existing outlier-detection methods use one or more parameters”. This seems to be largely correct for the outlier clustering techniques that are based on k-means. Similarly, the algorithm KMOR proposed by Gan et al. [7] asks for two additional parameters, one of which is the maximum number of outliers. The algorithm ODC [1] has the “difference” between outliers and real data points as a parameter, while Neo-k-means [15] asks for two parameters α and β , which are also related to the assignment of data points to a cluster, i.e. the number of outliers.

The aim of this work is to find a way to remove noise points from the clusters without additional parameters. Having to set parameters will most likely prevent inexperienced users from using data mining techniques, hence, the ideal clustering technique is one completely without parameters. One could argue that the other problems of k-means might have already deterred possible users, but these problems have many solutions, many of which are implemented in publicly available software. It is easy to add the option of, say, X-Means [12] to estimate the value of k and k-means++ [2] to achieve a good initialization. The ideal solution would be to add the option to remove noise too, without setting additional parameters.

1.2 Contributions

We present here a k-means based technique that adapts and expands k-means to noisy data sets.

- The technique presented in this work, KMN, can efficiently remove noise from k-means clustering results without additional parameters.
- It does so, while being deterministic. For a k-means clustering result, the excluded noise points are always the same.
- KMN offers a higher independence from the chosen k for k-means. This work demonstrates in Sect. 3 how KMN fares for wrongly chosen values of k and shows the resilience it gives in relation to these values.
- It is fully compatible with other k-means-based techniques that aim to improve k-means like X-Means and k-means++.

2 The Algorithm

The main intention of this work is to present an approach to exclude noise from the clustering result of k-means, to which we refer to as KMN. The algorithm starts with a k-means clustering result and tries to locate the areas where noise is prevalent. K-means follows a centre-oriented approach, in which data points are assigned to the nearest centre. The centre is then updated as the mean of the assigned data points. This means that the closer a datapoint is to a centre, the more likely it is to be assigned correctly (provided that k-means is a fitting technique for the dataset and k is correctly selected). Or in other words: the closer a data point is to the centre, the more likely is it correctly assigned. The further away, the more likely is it that it should be assigned to another centre or regarded as noise.

The search for noise should therefore begin at the locations furthest from the cluster centres. The clusters in k-means are Voronoi cells and between two neighbouring Voronoi cells there is a hyperplane that separates them. At the intersections of these hyperplanes one will find the point farthest from the centres, as shown in Fig. 2. Let us call this point m .

In a d -dimensional data space $d + 1$ voronoi cells¹ determine such an intersection point. This point m is basically defined as the point in the data space, where $\|v_j - m\| = r$, $j \in \{1, 2, \dots, d + 1\}$, holds; v_j are the centres of the voronoi cells, i.e. the centres that k-means finds, r is the distance of m to the centres of the voronoi cells. If one of the voronoi cell centres had a distance to m unequal r , m would be assigned to the voronoi cell with the smallest distance.

When these intersections are found, the assumption is valid that the area of these intersections should be considered to contain noise data points. Retaining the spirit of k-means, the algorithm then simply opens a new Voronoi cell at the intersection, but one in which all data points within its boundaries are regarded

¹ For additional information on the concept of voronoi cells we refer to [13] or the Wikipedia article.

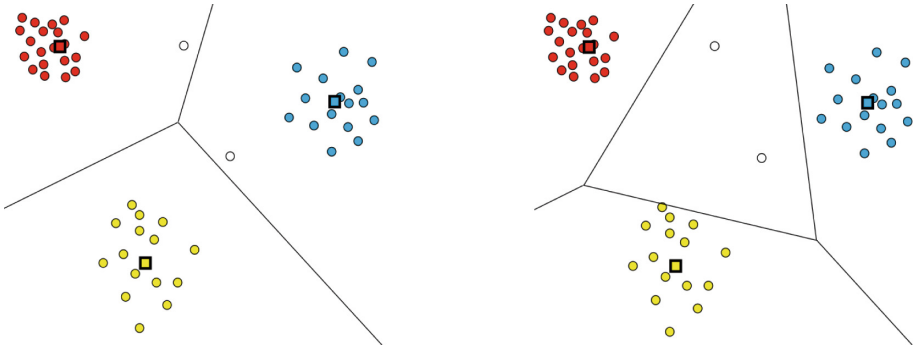


Fig. 2. Three simple clusters and two outliers. The squares are the centres of the clusters as found by k-means which determine the voronoi cell walls. This shows, if k-means is a suitable technique, that the data points near the centres are correctly assigned, while the more distant data points are more likely to be wrongly assigned. When a new Voronoi cell is opened at the intersection of Voronoi cells, the noise data points are separated from the clusters.

as noise (see Fig. 2(b)). It may not always be a good decision to open such a noise voronoi cell, because it is possible that such a noise voronoi cell contains data points and not (only) noise points. It is necessary to have a criterion to decide whether or not to open such a noise voronoi cell. KMN uses the principle of Minimum Description Length (MDL).

MDL is a well established principle in the Data Mining community and is used in various technique like X-Means [12]. The basic assumption behind MDL is that the coding cost for a clustering result depends on how good the clustering is. The coding cost is basically an estimate of how much memory is needed to encode the clustering. If the clustering is good, only little memory is needed, but if the clustering is far off, i.e. the model used to encode the data is not fitting, the encoding requires a lot of memory.

If the criterion is applied and the Voronoi cells are either retained or discarded, the next iteration can begin. We see in Fig. 2(b) that there are new Voronoi cell intersections and the same steps can be taken as before. Before this happens, the centres of the clusters are updated. The process is iterated until no more intersection improve the clustering, according to the criterion.

This is, very briefly, the procedure of the algorithm. Let us now elaborate on that.

2.1 Finding the Voronoi Intersections

There are several ways to find the intersections of the Voronoi cells. The most obvious way would be to find m using geometric calculations. In this approach one would to work with the equation $\|v_j - m\| = r$ directly. This equation is the formula for a sphere with radius r and midpoint m . The $d + 1$ points v_j are given and therefore uniquely determine the centre m . To calculate m , this would

entail inverting a $d + 1 \times d + 1$ -matrix, which would mean computations in the order of $O((d + 1)^3)$. While this might be acceptable, the difficulty lies with the choice of the v_j . If there are k centres of k-means in a d -dimensional data space, without knowing the adjacencies of the centres, one would have to compute all $\binom{k}{d+1}$ possible combinations to find the correct combinations of voronoi cells. This is comparatively expensive and should be avoided.

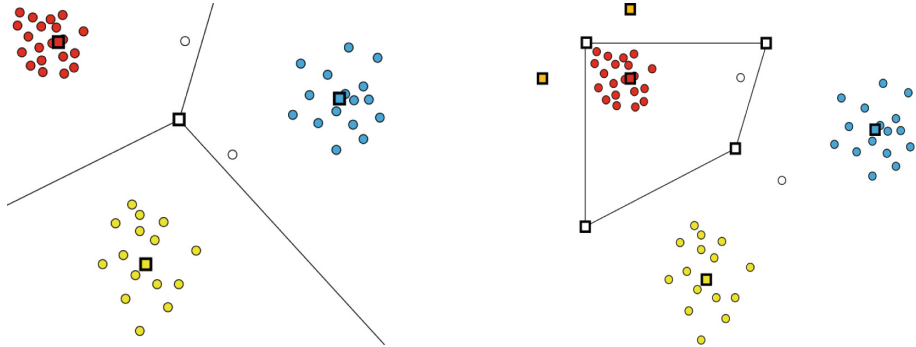


Fig. 3. Convex Hull and intersection. The three clusters determine one voronoi cell intersection, which is represented by the white square. The centres of the other clusters as well as the projections of its own centre determine a bounded polytope, i.e. the voronoi cell.

This technique uses the Avis-Fukuda algorithm [3], which is specialized in finding Voronoi intersections. More precisely, Avis-Fukuda takes the centre of each Voronoi cell, i.e. the centres that k-means finds, as input and calculates the intersection points of these Voronoi cells. The algorithm is based on a linear optimization approach. It works according to the following principle: it starts with a centre of a Voronoi cell and the other centres that bound it. This forms a convex hull around the centre, a polyhedron, with the corners of this hull being the intersections we are looking for. It calculates the nearest vertex with a linear optimization approach and then follows the beams of the convex hull along to find the other vertices. Avis-Fukuda follows the optimal path and finds all nodes, i.e. all Voronoi intersections. The algorithm uses “Bland’s rule” to ensure that the path is the optimal one and that all vertices are found. In this way, all corner points of a convex hull, i.e. all Voronoi intersections, are found.

A very simple example of this is illustrated in Fig. 3. The point of intersection between the three Voronoi cells is easy to find, but the Voronoi cell is unbounded. Therefore, we project the centre for all Voronoi cells beyond the data set boundaries, so that all clusters have a limited convex hull defined by the centres found by k-means and the added projections. The Avis Fukuda algorithm can then be used to find the vertices (represented in Fig. 3(b) as white squares). The algorithm continues by finding such a vertex and then moving on to the others. In a two-dimensional environment like this, it is rather straightforward. After the

first step, there is always only one direction in which the algorithm can advance. It follows the rays of the convex hull until it reconnects with the first vertex. The Avis-Fukuda algorithm has then found all nodes, i.e. all Voronoi intersections for the cluster.

The Avis-Fukuda algorithm is estimated to have a runtime of $\mathcal{O}(k \cdot d \cdot v)$, where v is the number of vertices. Since we apply the algorithm for each of the k voronoi cells, we have a runtime of $k \cdot \mathcal{O}(k \cdot d \cdot v)$.

2.2 The MDL-Criterion

The algorithm has found the possible intersections of the Voronoi cells. However, the question now arises as to whether the algorithm should open new Voronoi noise cells or not. MDL assumes that lower coding costs imply a better clustering. Thus, if the new Voronoi noise cell reduces the total coding costs, the algorithm keeps the new Voronoi noise cell. The coding costs consist of two parts: The coding costs for the model $L(M)$ and the coding costs for the data $L(D|M)$. The total coding cost is therefore $L(M, D) = L(M) + L(D|M)$. K is the number of clusters C_i ; N is the number of data points x ; p_i is the number of parameters. Hence, coding cost is given by the following equation:

$$\begin{aligned} L(M, D) &= L(M) && + L(D|M) \\ &= \sum_{i=1}^K \sum_{j=1}^{|C_i|} \log_2\left(\frac{N}{|C_i|}\right) + \sum_{i=1}^K \frac{p_i}{2} \log_2(|C_i|) && - \sum_{i=1}^K \sum_{x \in C_i} \log_2(\text{pdf}(x)) \end{aligned}$$

The model coding costs $L(M)$ is not difficult to calculate. Basically, one only needs to know the cluster sizes. The data encoding cost $L(D|M)$ is the more cumbersome part, since one needs to know the distribution of the data points. For algorithms like DBSCAN [6] that are not based on a probability distribution, this can be difficult, but k-means is based on the assumption of a Gaussian distributed cluster, where the variance is the same in all directions. Therefore we assume the data points to be Gaussian distributed in a cluster. Since the variance is the same in all directions, the distance of a data point from the centre is sufficient to determine its probability. We need two parameters to calculate the normal distribution of a cluster, the expected value and the variance/standard deviation of the cluster. The expected value is easy to calculate, it is simply the centre of the cluster. The variance is a slightly more difficult one.

Let x be a random point in the Gaussian cluster. We assume the cluster to be centred at 0. Every axis of the cluster is $N(0, \alpha)$ distributed, i.e. normal distributed with variance α , and we want to find α as it is the variance we are looking for. We calculate the distance of x to the centre $dist(x, 0) = \sqrt{\sum_{i=1}^d (\alpha X_i)^2}$, with X_i being a $N(0, 1)$ distributed random variable, so αX_i is $N(0, \alpha)$ distributed. We have:

$$dist(x, 0) = \sqrt{\sum_{i=1}^d (\alpha X_i)^2} = \sqrt{\alpha^2 \sum_{i=1}^d (X_i)^2} = \alpha \sqrt{\sum_{i=1}^d X_i^2}$$

The term $Y = \sqrt{\sum_{i=1}^d X_i^2}$ is known in the literature (e.g. [8]) as being Chi-distributed and with that we have its probability distribution, which we label as $pdf(x)$. The difference is that we have $\alpha\sqrt{\sum_{i=1}^d X_i^2}$, but probability theory tells us, if $\sqrt{\sum_{i=1}^d X_i^2} \sim pdf(x)$, then $\alpha\sqrt{\sum_{i=1}^d X_i^2} \sim \frac{1}{|\alpha|}pdf(\frac{x}{|\alpha|})$. We now know the form of the probability distribution.

The formula for the variance is $Var[Y] = E[Y^2] - E[Y]^2$. We can rewrite $E[Y^2]$ to $E[Y^2] = E[(\sqrt{\sum_{i=1}^d (\alpha X_i)^2})^2] = \alpha^2 E[\sum_{i=1}^d X_i^2]$. The literature tells us $Y^2 = \sum_{i=1}^d X_i^2$ is Chi-squared distributed and has a mean of d . Hence:

$$Var[Y] = \alpha^2 d - E[Y]^2$$

$$\alpha = \sqrt{\frac{Var[Y] + E[Y]^2}{d}}$$

The variance and mean values are the variance and mean of the distances of all data points to the centre, therefore we can compute them directly and get α .

$$pdf(x) = \frac{x^{d-1} e^{-\frac{x^2}{2\alpha^2}}}{2^{\frac{d}{2}-1} \alpha^d \Gamma(\frac{d}{2})}$$

is therefore the probability density we were looking for. Γ is the standard Gamma-function.

One could also estimate the variance differently, e.g. like X-Means [12], but this approach seems to be more compatible with the heuristics of our approach. Both have the same mathematical validity, but the use of the chi distribution for the pdf seems to take better account of the distortion of Gaussian spheres by outliers in the dataset. With the probability density function found, the coding costs of the data can be calculated and the MDL criterion is set up to test the Voronoi intersections.

The question that remains is which PDF should be used to model the noise. The obvious notion is to assume uniformly distributed noise, but this has the disadvantage of being massively distorted by outliers. Assume that the data set is completely within the $[0,1]^d$ -cube. A single data point $(2,2, \dots, 2)$ would change the volume of the data set from 1 to 2^d and thus also change the probability of a noise data point by a factor of 2^d . To make it more resistant to such extreme outliers, the algorithm assumes that the noise is also Gaussian distributed. The parameters for the noise distribution are calculated as before, whereby the mean value is the average value of all data points and the variance of the noise is the variance of all data points.

Let us go through the steps of the algorithm so far with Fig.4. We have the old Voronoi noise cells in white that are given by the centres of k-means and the projections of the cluster centres. They now determine the Voronoi cell intersections that are shown in gray. The MDL criterion is used to check whether the new Voronoi noise cells are to be opened at these intersections. Three of the four are accepted because they either do not change or reduce the coding cost.

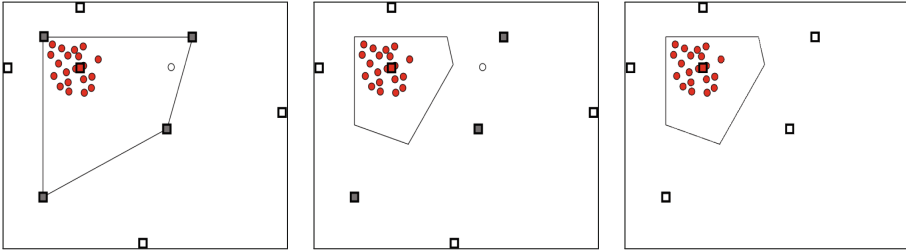


Fig. 4. The convex hull of a cluster in the course of one iteration. First, all potentially neighbouring Voronoi cells are determined (here only represented by their centres) and the intersection points of the cells are calculated; then those intersection points that open “good” noise cells, according to the MDL principle, are retained. Finally, those noise cells that are no longer adjacent are removed.

The fourth option would, however, massively increase the coding cost as it would move many data points from the cluster to a noise cell. Therefore, only three of these intersections are kept and the new convex hull of the cluster is described by the Voronoi cells, which centres can be seen in Fig. 4(b). One can see that not all these centres are necessary any more, since two of the old centres are no longer adjacent to the Voronoi cell. When eliminated, the situation would be as shown in Fig. 4(c).

If one were to keep these unnecessary noise cells, the next iteration would take slightly longer and the next one somewhat longer. Depending on the number of iterations the runtime would increase massively, so the question arises how to eliminate the unnecessary centres.

2.3 Finding the Voronoi Adjacencies

The Avis-Fukuda algorithm offers the possibility to calculate the Voronoi cell adjacencies, but is not focused on this by default. Mendez et al. [11] created an algorithm based on linear optimization, which specializes in this.

The algorithm translates the Voronoi adjacency problem into the linear optimization version of it and then continues the dual problem as it is more practical to solve. It begins with the equivalent of two randomly selected centres of Voronoi cells and tests whether the point in the middle is located in one of the two Voronoi cells. If this is the case, the Voronoi cells are adjacent to each other. If this is not the case, the dual problem forms the basis for the linear optimization problem, which could contain information about the neighbourhood of several other Voronoi cells. Therefore, the algorithm is faster than the Avis Fukuda algorithm, which calculates the adjacency of all pairs of Voronoi cells (see [11] for details). Mendez et al. estimate the runtime of their algorithm as $\mathcal{O}(k^2 \cdot f(k, d))$, where f is unknown, but not worse than polynomial.

By using the Mendez algorithm, our technology can now eliminate the unnecessary centres for our problem. The next iteration would now start with the

centres shown in Fig. 4(c) and repeat the same steps as before. The algorithm updates the centres of the clusters in the same way as k-means, resulting in a small change of the Voronoi cell structure. If the algorithm rejects all new possible Voronoi noise cells of an iteration, then it is converged and the cluster is marked as “fully cut out”.

Algorithm 1. KMN

```

Require: k-means clustering result  $D$ 
1: procedure KMN( $D$ )
2:   Initialize: Compute initial voronoi cell adjacencies ▷  $\mathcal{O}(k^2 \cdot f(k, d))$ 
3:   while Coding cost decreases do ▷  $l$  times
4:     Find  $v$  voronoi cell intersections ▷  $k \cdot \mathcal{O}(k \cdot d \cdot v)$ 
5:     Check intersections  $v$  with MDL ▷  $\mathcal{O}(v \cdot n)$ 
6:     Compute adjacencies ▷  $\mathcal{O}(k^2 \cdot f(k, d))$ 
7:     Update cluster centres ▷  $\mathcal{O}(k \cdot n)$ 
8:     Compute coding cost ▷  $\mathcal{O}(n)$ 
9:   end while
10:  return Cluster  $C_1, \dots, C_k$  and Noise  $N$ 
11: end procedure

```

2.4 Pseudo Code

Following the pseudocode shown in Algorithm 1 we can assume that the runtime of this approach is in the order of $\mathcal{O}(l \cdot k^2 \cdot d \cdot v^2 \cdot n \cdot f(k, d))$. This shows us that this approach is relatively stable in terms of the data size n , but is somewhat more influenced by the dimensionality d and the number of clusters k . This is logical because the algorithm has to calculate the Voronoi cell intersections and adjacencies. This is independent of the size of the data set itself and could also be seen as a constant; the algorithm itself would then have a purely linear dependency on n , like k-means itself has.

2.5 Performance on Running Example

The theory behind this technique has been presented, now let us see how it performs on our running example. In Fig. 5a we have the result of k-means on a simple dataset, consisting of 5 clusters and 10% noise. The clusters are Gaussian distributed, as k-means assumes, and differ quite strongly in density and size. The data set is well suited for k-means and all clusters are well separated by it. The main error lies in the wrongly assigned noise points.

The algorithm iteratively computes the intersections, tests them with the MDL criterion, excludes noise points and updates the centres. The convex hulls shrink until no more Voronoi noise cells are kept. For some of the clusters, this happens earlier than for others due to their spread. At the end almost all noise

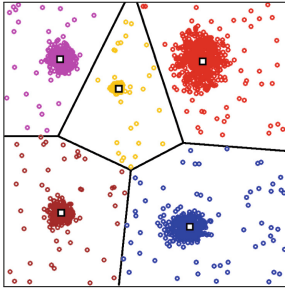
points are found and the clusters are cut out near ideally. The improvement can also be measured with the help of the “normalized mutual information”-measure (NMI) [14]. The NMI value increases from 0.82 for the k-means result to 0.94 for the result of our approach. Some of the outliers are located in the middle of a cluster and therefore cannot be recognized as such. Nevertheless, they are considered to be outliers. Therefore, a perfect NMI of 1.0 is impossible to obtain and 0.94 is almost the best result one can get.

3 Resilience Regarding k

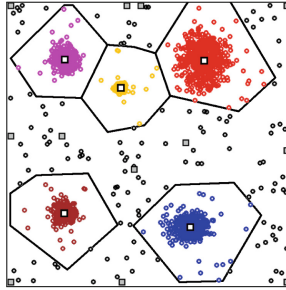
This technique has another advantage, which we would like to present now. For regular k-means, setting k is one of the most important decisions and k is often difficult to estimate. Tools like X-Means [12] help the user with this decision, but are not necessarily correct. KMN now gives some leeway for this decision. We see this in our current example with a wrong k . If we have chosen a value of k that is too small, e.g. $k = 3$, k-means can by default not separate all clusters. The clustering result will necessarily look similar to the one in Figure 6(a), where two clusters are merged into one. KMN now has the advantage that it eliminates data points from a cluster if they do not fit. We see the result in the first iteration. The sub-clusters were (mostly) excluded and added to the noise points. The centre is updated (we see that it moves from Fig. 6(b) and (c)) and moves to one of the correct clusters in the following iterations. At the end (Fig. 6(d)) three of the five clusters are found and separated from the noise, while two of them are simply added to the noise.

This resilience is also supported in the opposite direction. In Fig. 7(a) we have k set to 6. Therefore, k-means assigns some of the noise points to a separate cluster consisting only of noise points. We see that this “wrong” cluster loses some of its data points in the following iteration (Fig. 7(b)). It decreases from iteration to iteration until it is practically emptied (Fig. 7(d)). KMN has reduced the value of k from 6 to 5 by declaring a cluster as completely empty. We see that this is no coincidence if we choose an even bigger k of 10. The result is shown in Fig. 8(a), (b) and (c). K-means has found three of the correct clusters (with some extra noise), three clusters consisting of pure noise and divided two clusters into half. When KMN is now applied to this result, the three correct clusters iteratively lose noise and converge to their correct shape. The clusters, which are divided into two halves, also lose their noise points, but remain divided into two halves. KMN does not currently have a function to merge clusters, but we hope to extend KMN to do so in the future. The three “false” clusters that consist only of noise, get most of their data points reassigned to noise. In the end, they consist of no or almost no data points. They effectively disappear.

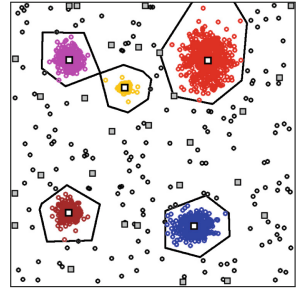
We see that in the end, KMN may not be a technique for correctly estimating k , but it gives quite some leeway for the correct estimation. This does take some pressure from the estimation of k and techniques like X-Means are given a wider range of correct values.



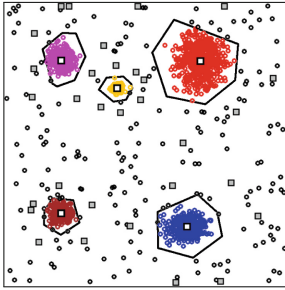
(a) K-means with $k=5$. NMI is 0.82.



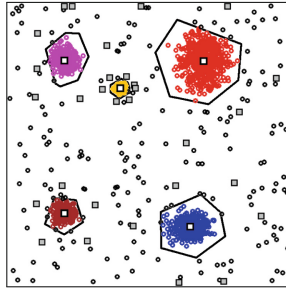
(b) The first iteration



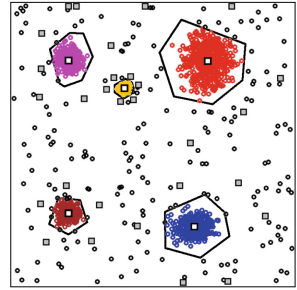
(c) Most noise is found after the second iteration.



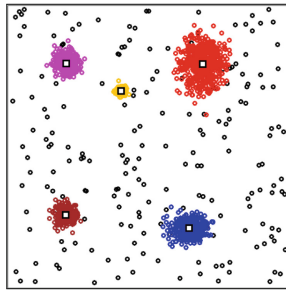
(d) The convex hulls have clearly shrunk.



(e) Some of the convex hulls do not change any more.



(f) The final result. The clusters are separated from the noise. NMI is 0.94.



(g) Ground truth.

Fig. 5. The stages of KMN from the result of k-means to the final clustering. Cluster are displayed in different colours. Cluster-centre as white squares, centre of adjacent Voronoi cells with grey squares. The last Figure shows the ground truth. Figures best viewed in colour. (Color figure online)

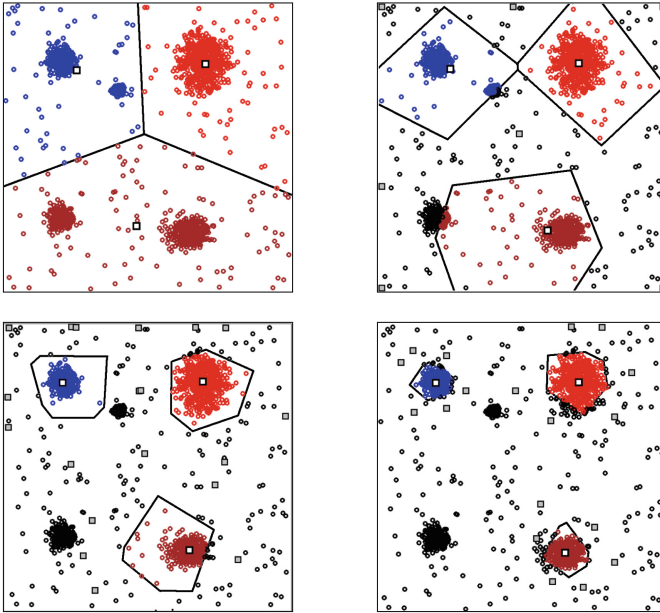


Fig. 6. The behaviour of the algorithm for a too small value of $k = 3$. Depicted is the result of k-means, the state after the first iteration, the second iteration and the final result.

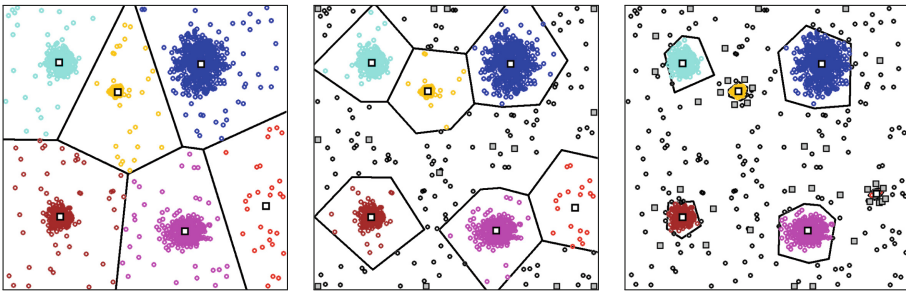


Fig. 7. K is here set to 6. Depicted is the result of k-means, the state after the first and the second iteration, as well as the final result.

4 Experiments

Synthetic Data. We have compared the algorithms on the running example and the clustering results are shown in Table 1. The reason for using NMI as a measure for clustering lies in the opinion to see KMN as a clustering-in-the-presence-of-noise-technique. k-means-- has been given the correct value for the outliers, for Neo-k-means we used the internal estimator for the parameters. If a data point was assigned to more than one cluster by Neo-k-means, it was

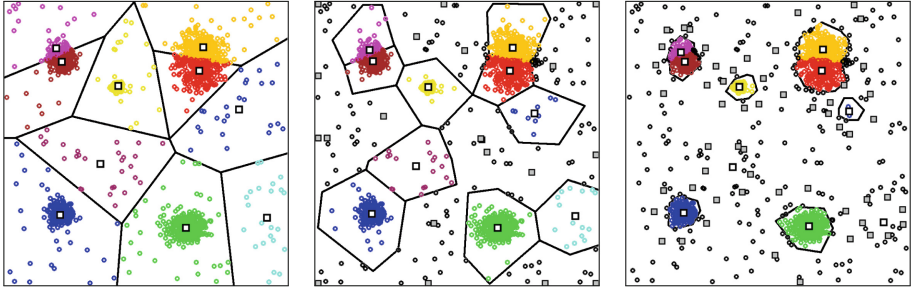


Fig. 8. K is here set to 10. Depicted is the result of k-means, the state after the first iteration and the final result.

assigned to the nearest centre. Each experiment was repeated 50 times and the average value is displayed in Table 1. We can see that KMN clearly outperforms k-means-- and Neo-k-means if the correct value of k is specified, but even if k is off.

Table 1. NMI Values for our running example for varying values of k .

	k=5	k=2	k=3	k=4	k=6	k=10
KMN	0.874	0.412	0.675	0.813	0.850	0.780
k-means--	0.824	0.410	0.595	0.749	0.816	0.712
Neo-k-means	0.760	0.354	0.546	0.713	0.808	0.722
k-means	0.765	0.352	0.554	0.708	0.793	0.755

Real World Data. The difficulty of testing outlier detecting cluster techniques lies in the lack of suitable datasets, i.e. datasets containing noise data points. Campos et al. have compiled a list of possible datasets that can be used for such techniques [4]. Most of these are UCI [9] datasets for which some of the classes have been declared noise. Most of these are also somewhat unsuitable for k-means-based techniques, since the cluster results of k-means have a very low quality (i.e. very low NMI). So there are very few datasets that we can use. Due to this, we have also included another UCI dataset showing the behaviour of the algorithm on an outlier-free datasets.

Table 2. NMI Values for real word data sets.

	Hayes-Roth	Glass	WBC
KMN	0.10	0.34	0.56
k-means--	0.08	0.33	0.34 (l= 120) 0.78 (l= 241) 0.45 (l= 361)
Neo-k-means	0.08	0.34	0.00

The first dataset in Table 2, Hayes-Roth, is without outliers. The outlier parameter of `k-means--` was therefore set to 0, which means that `k-means--` gives the same results as `k-means`. Therefore, we see that KMN improves on the result of `k-means`. This is because `k-means` assigns some data points to the wrong cluster, which KMN finds and identifies as noise. KMN notices that they do not belong in the current cluster. Clustering noise-free datasets often yields a small improvement in clustering quality compared to `k-means`. Not enough to use KMN on a data set, where noise is known not to be present, but enough to warrant mention.

The data sets Glass and WBC both contain outliers. One has to keep in mind that `k-means--` needs to know the number of outliers, which is often very difficult to estimate, while KMN is parameter-free. On the WBC data set `k-means--` fares better when given the correct number of outlier. The NMI-values become identical when the outlier-number is off by roughly 30% and when the value is off by more than that, KMN delivers the better results.

The data sets were each clustered 50 times per algorithm. The runtime of KMN for WBC proved to be quite high and hence only one iteration was performed. All cluster-algorithms were given the correct values of k on the data sets.

5 Outlook and Conclusion

We wanted to create an algorithm that would be able to remove noise data points from a `k-means` clustering and could achieve this without any additional parameters. In Fig. 1 we see how much a `k-means` clustering result can deviate from the correct clustering, even though the data set is well suited for `k-means`, simply due to the noise data points that `k-means` cannot account for. Through KMN we have now developed an additional algorithm for `k-means`, which can be used to remove noise data points. Due to its modular design it can be used after `k-means` has run its course, which means that it is completely compatible with other extensions of `k-means`, such as `k-means++`. Moreover, we were also able to show that KMN makes `k-means` more robust in terms of too small or big k value.

For future work we have the goal to extend KMN towards a general noise extracting algorithm that can be applied as an addition for any clustering algorithm. For this goal it is necessary to abolish the voronoi cell structure of this algorithm, since that is inherent for `k-means`, but not necessarily for other algorithms. Removing the voronoi cell structure might also lead to a more dynamic approach that would give us a greater flexibility.

References

1. Ahmed, M., Naser, A.: A novel approach for outlier detection and clustering improvement. In: ICIEA (2013)
2. Arthur, D., Vassilvitskii, S.: `k-means++`: the advantages of careful seeding. In: SODA (2007)

3. Avis, D., Fukuda, K.: A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discret. Comput. Geom.* **8**, 295–313 (1992)
4. Campos, G., et al.: On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data Min. Knowl. Discov.* **30**, 891–927 (2016)
5. Chawla S., Gionis A.: k-means–: a unified approach to clustering and outlier detection. In: *ICDM* (2013)
6. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: *KDD* (1996)
7. Gan, G., Kwok-Po Ng, M.: k-means clustering with outlier removal. *Pattern Recognit. Lett.* **90**, 8–14 (2017)
8. Johnson, N., Kotz, S., Balakrishnan, N.: *Continuous Univariate Distributions*. Houghton Mifflin, Boston (1994)
9. Lichman, M.: *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences (2013)
10. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: *Berkeley Symposium on Mathematical Statistics and Probability* (1967)
11. Mendez, J., Lorenzo, J.: computing voronoi adjacencies in high dimensional spaces by using linear programming. In: Latorre Carmona, P., Sánchez, J., Fred, A. (eds.) *Mathematical Methodologies in Pattern Recognition and Machine Learning*. Springer Proceedings in Mathematics & Statistics, vol. 30, pp. 33–49. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-5076-4_3
12. Pelleg, D., Moore A.W.: X-means: extending K-means with efficient estimation of the number of clusters. In: *ICML* (2000)
13. Preparata, F., Shamos, M.: *Computational Geometry: An Introduction*. Springer, New York (1985). <https://doi.org/10.1007/978-1-4612-1098-6>
14. Vinh, N.X., Bailey, J.: Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance. *JMLR* **11**, 2837–2854 (2011)
15. Wangh, J.J., Dhillon, I., Gleich, D.: Non-exhaustive, Overlapping k-means. In: *SDM* (2015)



Subset Labeled LDA: A Topic Model for Extreme Multi-label Classification

Yannis Papanikolaou^(✉) and Grigorios Tsoumakas

School of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece
{ypapanik,greg}@csd.auth.gr

Abstract. Labeled Latent Dirichlet Allocation (LLDA) is an extension of the standard unsupervised Latent Dirichlet Allocation (LDA) algorithm, to address multi-label learning tasks. Previous work has shown it to perform en par with other state-of-the-art multi-label methods. Nonetheless, with increasing number of labels LLDA encounters scalability issues. In this work, we introduce Subset LLDA, a topic model that extends the standard LLDA algorithm, that not only can efficiently scale up to problems with hundreds of thousands of labels but also improves over the LLDA state-of-the-art in terms of prediction accuracy. We conduct experiments on eight data sets, with labels ranging from hundreds to hundreds of thousands, comparing our proposed algorithm with the other LLDA algorithms (Prior-LDA, Dep-LDA), as well as the state-of-the-art in extreme multi-label classification. The results show a steady advantage of our method over the other LLDA algorithms and competitive results compared to the extreme multi-label classification algorithms.

Keywords: Machine learning · Extreme classification · Topic models
Multi-label classification

1 Introduction

Multi-label learning addresses supervised learning problems, where each training example is associated with more than one labels at the same time [12]. Examples include tasks such as image annotation or assigning concepts to documents. Although a great body of prior work has dealt with developing methods for multi-label tasks (e.g. [16]), the majority of these algorithms struggle to scale to data sets having more than a few thousand labels. The ever increasing flow and volumes of data in modern-day applications call for multi-label learning algorithms that can scale up effectively and efficiently.

Extreme multi-label classification (XMLC) is an emerging field that attempts to address the above challenge, by proposing algorithms that can tackle problems with extremely large label sets ($> 10^4$ labels).

We modify an already existing algorithm, Labeled Latent Dirichlet Allocation (LLDA) [8] to successfully deal with such tasks. LLDA was introduced as an extension of standard, unsupervised Latent Dirichlet Allocation (LDA) [2], to

deal with multi-label learning tasks. Apart from delivering results competitive with state-of-the-art multi-label algorithms, LLDA’s training is by design fit for large-scale and extreme learning problems, since its training time complexity is not dependent of the label set size L , but on the average number of labels assigned per instance. During testing though, LLDA reduces to LDA and the algorithm is linearly dependent of L , which makes it unfit for XMLC.

In order to make LLDA appropriate for such tasks, we propose an extension to LLDA, Subset LLDA. Our algorithm is different only during prediction. It first determines a set of relevant labels for each new instance, through its nearest neighbors in the training space, and then constrains the LLDA’s inference on this particular subset of labels. By doing so, we manage to decrease the algorithm’s testing time complexity and improve LLDA’s quality of predictions, since by constraining the algorithm to search in a subset of the entire label space, we alleviate LLDA’s tendency to converge to local optima (we describe this more in detail in Sect. 3).

We conduct experiments on four small scale data sets and four large scale data sets, with L ranging from 101 to 670,000 comparing our approach to Prior-LDA and Dep-LDA as well as two of the top performing extreme classification algorithms, Fast XML and PfastreXML. The results show a consistent advantage of our method compared to the other LLDA algorithms and competitive results with the extreme classification methods. Our motivation by introducing Subset LLDA, is to contribute one more method to the extreme classification inventory, that may be more apt than other methods for specific experimental scenarios.

2 Background and Related Work

We present here the main methods in the literature to tackle XMLC tasks and then present LDA, LLDA, and the other two LLDA extensions, Prior-LDA and Dep-LDA. Throughout the paper we assume that the Collapsed Gibbs Sampling (CGS) algorithm [4] is employed for all LLDA methods.

2.1 Extreme Classification Methods

Algorithms aiming to tackle extreme classification tasks mainly take one of the following approaches:

- learn a hierarchy out of the training set, either over the labels or the features, and solve the training and prediction procedures locally at each node. Examples in this category include Label Partitioning by Sublinear Ranking (LPSR) [13], FastXML [7] and PfastreXML [5].
- construct an embedding of the output space in a lower dimension. Embedding-based methods render training and prediction tractable by assuming that the training label matrix is low-rank, reducing the label set size by projecting the high dimensional label vectors onto a low dimensional linear subspace. Most characteristic methods in this category are LEML [15] and SLEEC [1].

2.2 LDA and LLDA

Let us denote as L the number of labels, l being a label and V the number of features, v being a feature type and w_i being a feature token at position i of the instance. M is the number of instances (M_{TRAIN} and M_{TEST} will represent the training and testing set sizes respectively), m being an instance. Also, L_m will denote the number of m 's labels and N_m the number of its non-zero features.

LDA assumes that, given a set of instances, there exist two sets of distributions, the label-features distributions named ϕ and the instances-labels distributions named θ ¹. CGS LDA marginalizes out ϕ and θ , and uses only the latent variable assignments \mathbf{z} . The algorithm employs two count matrices during sampling, the number of times that v is assigned to l across the data set, represented by n_{lv} and the number of feature tokens in m that have been assigned to l , represented by n_{ml} . During sampling, CGS updates the hard assignment z_i of w_i to one of $l \in \{1 \dots L\}$. This update is performed sequentially for all tokens in the data set, for a fixed number of iterations. The update equation giving the probability of setting z_i to label l , conditional on w_i , m , the hyperparameters α and β , and the current label assignments of all other feature tokens (represented by \cdot) is:

$$p(z_i = l | w_i = v, m, \alpha, \beta, \cdot) \propto \frac{n_{lv \rightarrow i} + \beta}{\sum_{v'=1}^V (n_{lv' \rightarrow i} + \beta)} \cdot \frac{n_{ml \rightarrow i} + \alpha_l}{N_m + \sum_{l'=1}^L \alpha_{l'}}. \quad (1)$$

Upon completion of the above procedure, point estimates can be calculated for ϕ and θ parameters. Instead of the standard CGS estimators [4], we employ the *CGSP* equations [6], that employ the full distribution over feature tokens. These methods calculate the expected values of the standard CGS estimators and are therefore especially suited for the large-scale setting that we are addressing in this work, since they allow us to achieve better performance by drawing fewer samples than with the standard estimators.

LLDA modifies LDA by constraining the possible assignments for a token to a label to the instance's observed labels. Inference on test instances is performed similarly to unsupervised CGS - LDA: the label-features distributions, ϕ , are fixed to those previously learned on the training data, and the test instances' θ distributions are estimated.

LLDA employs a symmetric α hyperparameter over labels, giving equal weight on them. Nevertheless, in most real-world tasks labels tend to have skewed distributions. Moreover, modeling label dependencies can improve performance [9], especially as the number of labels increases. To address these issues, the authors of [10] have proposed Prior-LDA and Dep-LDA respectively. Prior-LDA incorporates the label frequencies observed in the training set via an asymmetric

¹ Specifically, LDA is defining ϕ and θ in terms of topics since it is unsupervised, but to ease understanding we consider through the paper that topics and labels are equivalent.

α hyperparameter: a frequent label will have a larger α'_l value than a rare one. Specifically, it is set to

$$\alpha'_l = \eta \cdot f_l + \alpha \quad (2)$$

with η , α being user defined parameters and f_l representing the frequency of l in the training corpus, $f_l \in [0, 1]$.

Dep-LDA is a two-stage algorithm: first, an unsupervised LDA model is trained with T topics and using as training data, each instance’s label set². The estimated LDA model will incorporate information about the label dependencies, since relevant labels will tend to be described by the same topic(s). Second, an LLDA model is trained. During prediction, the previously estimated θ' , ϕ' parameters of the unsupervised LDA model are used to calculate an asymmetrical α'_{ml} . Specifically, the α'_m vector will be

$$\alpha'_m = \eta(\theta'_m \cdot \phi') + \alpha \quad (3)$$

3 Subset LLDA

When dealing with tasks with very large L ($> 10^4$), LLDA and its extensions can not scale in a satisfying manner since they are linearly dependent on L during prediction. To alleviate this, we propose a simple extension to standard LLDA during prediction, by constraining the label space in which the algorithm can search for solutions. Specifically, our method proceeds in two stages:

- First, for each test instance, a set of candidate labels, denoted as $\mathcal{L}_{m_{Rel}}$, is determined. A number of approaches can be followed to determine this candidate list, for simplicity we retrieve the n -nearest instances from the training set, denoted as $\mathcal{M}_{m_{Rel}}$, and set the candidate list as the union of the retrieved instances tags.
- Second, we predict with LLDA, but constrain the possible labels to $\mathcal{L}_{m_{Rel}}$. Similarly to Prior-LDA, we modify the prior on the instance-topics distributions to reflect the frequencies of the labels among the n -most similar instances. For instance, if $n = 10$ and a given label l_A has appeared in three of the similar instances, while a label l_B has appeared in five of the similar instances, we set $\alpha'_{l_A} = \eta + 0.3 \cdot \alpha$, $\alpha'_{l_B} = \eta + 0.5 \cdot \alpha$.

Formally, our topic model makes the following assumptions: First, given an instance m and given a set of already tagged instances \mathcal{M}_{Train} , m ’s label set \mathcal{L}_m will be included in the union of the n most similar instances from \mathcal{M}_{Train} . Clearly, that assumption holds always as $n \rightarrow M_{Train}$ but in any other case it will represent an approximation and we expect that $\mathcal{L}_m \subseteq \mathcal{L}_{m_{Rel}}$. A second assumption relates to each label’s weight during Gibbs sampling: we hypothesize that for a given instance m , its labels have been generated by sampling from a multinomial distribution ϕ'_m . This assumption is very similar to the one employed in [10], when introducing Prior-LDA, except that we constrain the set from which the instance’s labels are generated to $\mathcal{L}_{m_{Rel}}$, instead of \mathcal{L} . Figure 1 illustrates our proposed model in graphical model notation.

² i.e., the feature tokens of each instance are its labels.

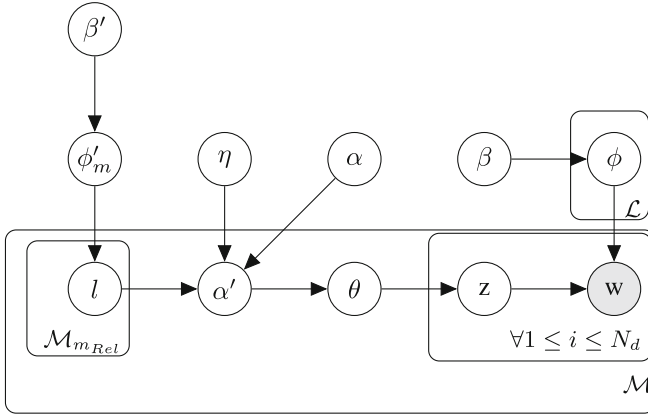


Fig. 1. Subset LLDA in graphical model notation. Our model assumes that, given an instance m , its labels have been generated by sampling from a multinomial distribution over m 's similar instances.

From the above, the generative process for Subset LLDA will be:

- For each $l \in \mathcal{L}$, sample a distribution $\phi_l \sim \text{Dirichlet}(\beta)$ over \mathcal{V}
- for each instance m
 - Sample n instances from \mathcal{M}_{Train}
 - Set $\mathcal{L}_{m_{Rel}} = \bigcup \mathcal{L}_{m_i}$ with $m_i \in \mathcal{M}_{m_{Rel}}$ and $i \in \{1..n\}$
 - Sample a multinomial distribution $\phi'_m \sim \text{Dirichlet}(\beta')$
 - Calculate α' according to Eq. 2.
 - Sample a distribution $\theta_m \sim \text{Dirichlet}(\alpha')$ over $\mathcal{L}_{m_{Rel}}$
 - For each feature position $i \in \{1..N_m\}$
 - * Sample a label $z_i \sim \text{Multinomial}(\theta_m)$
 - * Sample a feature type $v_i \sim \text{Multinomial}(\phi_l)$ with $l = z_i$.

Constraining the label set during prediction can also be useful for an additional reason. In that phase, LLDA needs to search the entire label space to recommend labels for a given test instance. Since this is a probabilistic method, the algorithm may converge to local optima, especially as L is increasing. To concretize, let us consider a trained LLDA model for which a specific feature v has a high probability ϕ_{lv} for several labels. Also, let us consider a test instance m that contains v , for which only one of the aforementioned labels is semantically relevant. In that case, it is possible that these noisy labels, coupled with LLDA's probabilistic nature, will lead the algorithm to favor one of the irrelevant labels at the expense of the correct label. This problem can of course be relieved by averaging over many samples and Markov Chains (MC) [6, 10], but in most real cases this is too expensive time-wise.

Finally, we note that to retrieve the most similar training instances we use the tf-idf representation for each instance and employ the cosine similarity, setting $n = 10$.

Table 1. Statistics for the data sets used in the experiments.

Data set	D_{Train}	D_{Test}	L	V
Bibtex	4,880	2,515	159	1,836
Delicious	12,910	3,181	983	500
Mediamill	30,993	12,914	101	120
EUR-Lex	15,539	3,809	3,993	5,000
Wiki10	14,146	6,616	30,938	101,938
BioASQ	40,000	10,000	19,218	36,480
Delicious-200k	196,606	100,095	205,443	782,585
Amazon-670k	490,449	153,025	670,091	135,909

3.1 Time Complexity

The CGS algorithm for LDA proceeds as follows: for every feature token of every instance in the corpus, it calculates probability distribution over all labels and then samples a label for the token, out of this calculated probability. In this way, standard LDA is linearly dependent on L . Formally, it will be

$$T_{LDA} \propto \mathcal{O}(M \cdot \overline{V}_m \cdot L). \quad (4)$$

LLDA, introduces supervision during training, by constraining the possible labels that a feature token can get on the instance’s label set L_d . Formally, during training LLDA’s complexity will be

$$T_{LLDA} \propto \mathcal{O}(M_{TRAIN} \cdot \overline{V}_m \cdot \overline{L}_m). \quad (5)$$

As explained, during testing LLDA is equivalent to LDA so its complexity will be given by Eq. 4. To alleviate this, with our approach we constrain Subset LLDA during testing, to only consider labels from the n -most similar instances. The total complexity of Subset LLDA, involves also finding the n -most similar instances which in our cases will be $\mathcal{O}(M_{TEST} \cdot M_{TRAIN})$:

$$T_{SubsetLLDA} \propto \mathcal{O}(M_{TEST} \cdot \overline{V}_m \cdot \nu \cdot \overline{L}_m + M_{TEST} \cdot M_{TRAIN}). \quad (6)$$

4 Empirical Evaluation

We here present the data sets, the setup and the results of the experiments that we carried out. We compare Subset LLDA with Prior-LDA, Dep-LDA, Fast XML and PfastreXML.

In our experiments, we employed four small scale and four large scale data sets, their statistics being illustrated in Table 1. The motivation behind using the four small scale data sets is to be able to compare our algorithm with the other LLDA variants, as well as to provide an empirical comparison against

Table 2. Micro-F and Macro-F results for the LLDA-based and the extreme classification methods. A ∇ indicates a statistically significant difference between Subset LLDA and the best performing method, with a z-test and a significance level of 0.05. The - sign is used if the algorithm could not deliver predictions after 48 hours.

(a) Micro-F

	FastXML	PfastreXML	PriorLDA	DepLDA	Subset LLDA
Bibtex	0.382	0.397	0.363	0.314	0.384
Delicious	0.347 ∇	0.322	0.255	0.273	0.304
Mediamill	0.577	0.572	0.511	0.512	0.580
EUR-Lex	0.413	0.436	0.321	0.357	0.443 ∇
BioASQ	0.382	0.370	-	-	0.428 ∇
Wiki10	0.317	0.33 ∇	-	-	0.283
Delicious-200k	0.162 ∇	0.129	-	-	0.135
Amazon	0.192	0.316 ∇	-	-	0.243
Avg. Rank	2.125	2.000			1.875

(b) Macro-F

	FastXML	PfastreXML	PriorLDA	DepLDA	Subset LLDA
Bibtex	0.273	0.288	0.257	0.204	0.292
Delicious	0.153	0.165 ∇	0.100	0.090	0.136
Mediamill	0.070	0.101 ∇	0.027	0.027	0.068
EUR-Lex	0.427	0.416	0.305	0.336	0.444
BioASQ	0.392	0.399	-	-	0.451 ∇
Wiki10	0.305 ∇	0.285	-	-	0.272
Delicious-200k	0.105 ∇	0.067	-	-	0.078
Amazon	0.426	0.483	-	-	0.486
Avg. Rank	2.125	2.000			1.875

the other extreme classification methods, in standard multi-label classification settings. All data sets apart from *BioASQ*³ [11] were retrieved from the extreme classification repository, with the respective training/testing splits.

Fast XML and PfastreXML were used with default parameters, with the relevant software packages provided in the extreme classification repository. For the LLDA models, we provide our Java implementation⁴. We trained the same model for all algorithms in order to ensure fairness of comparison, with $\alpha_l = \frac{50}{L}$. For Dep-LDA, we additionally need to train an LDA model to calculate the hyperparameter on θ (ref. to Eq. 3). For its training we use 100 topics and 200 iterations, 50 burn-in iterations and we set $\alpha = 0.1$, $\beta = 0.01$. During prediction, we set $\eta = 50$, $\alpha = \frac{30}{L}$, $\beta = 0.01$ for all LLDA models. In both training and prediction and across all data sets, we used one Markov Chain with 200 iterations and 50 iterations burn-in, averaging across samples to obtain the respective parameter estimates for each method. All algorithms output rankings of relevant

³ The exact data set used in the experiments is available upon request to the authors.

⁴ <https://www.dropbox.com/s/rypmlt18zk6jxdh/sllda.zip?dl=0>.

Table 3. Precision@1 and Precision@5 results for the LLDA-based and the extreme classification methods.

(a) Precision@1

	FastXML	PfastreXML	PriorLDA	DepLDA	Subset LLDA
Bibtex	0.645 ∇	0.565	0.551	0.501	0.579
Delicious	0.705 ∇	0.546	0.488	0.526	0.525
Mediamill	0.873 ∇	0.867	0.794	0.794	0.813
EUR-Lex	0.637	0.644	0.622	0.631	0.663 ∇
BioASQ	0.176	0.189 ∇	-	-	0.077
Wiki10	0.825 ∇	0.757	-	-	0.773
Delicious-200k	0.432 ∇	0.261	-	-	0.163
Amazon	0.286	0.368 ∇	-	-	0.350
Avg. Rank	1.625	2.000			2.375

(b) Precision@5

	FastXML	PfastreXML	PriorLDA	DepLDA	Subset LLDA
Bibtex	0.286 ∇	0.254	0.238	0.215	0.243
Delicious	0.596 ∇	0.482	0.391	0.416	0.427
Mediamill	0.531	0.534	0.480	0.471	0.539
EUR-Lex	0.425	0.445	0.332	0.358	0.457 ∇
BioASQ	0.170	0.169	-	-	0.174
Wiki10	0.570	0.572	-	-	0.560
Delicious-200k	0.362 ∇	0.262	-	-	0.201
Amazon	0.195	0.320 ∇	-	-	0.246
Avg. Rank	2.000	1.875			2.125

labels for each instance, so we used the *rcut* thresholding strategy [14] to compute the Micro-F and Macro-F scores.

4.1 Results

In Tables 2, 3 and 4 we report the results of our experiments. For the LLDA methods, we report the average over five runs.

First, let us consider the differences in prediction accuracy among the LLDA methods. Subset LLDA steadily outperforms both Prior-LDA and Dep-LDA in all settings and for all measures. It should be noted here, that Dep-LDA by design would benefit by averaging over more samples and more than one MC more than the other methods, since it employs the parameters learned from an unsupervised LDA model to calculate the hyperparameters for θ , therefore it will be more prone than the other algorithms to get stuck in local optima. In other words, the LDA model introduces an additional factor of uncertainty, which could be alleviated with more samples and chains. We nevertheless restrict our experiments to only one MC and relatively few samples (thirty) since our main goal is to address large-scale tasks in which multiple MC averaging and averaging over many samples is not feasible. One more interesting observation is that for

tasks with few labels (*Bibtex*, *Mediamill*), Dep-LDA performs equally or worse to Prior LDA which may be explained by the fact that modeling dependencies does not necessarily help improving performance in small scale tasks.

Comparing Subset LLDA with the extreme classification methods, we observe that different algorithms fare well in different evaluation measures. By considering the average rank per evaluation measure (last row of the tables), we observe that Subset LLDA achieves the first place for two measures, Micro-F and Macro-F, PfastreXML for precision@5 and FastXML for precision@1. FastXML and PfastreXML are better in predicting a few relevant labels per instance, while our algorithm is better in balancing precision and recall (through the F-measure) both when treating all labels equally (Macro-F) and when weighting them by their frequency (Micro-F).

Table 4. Training and prediction duration, in seconds, for the LLDA-based and the extreme classification methods.

	FastXML	PfastreXML	PriorLDA	DepLDA	Subset LLDA
Bibtex	3+1	3+1	29+73	32+64	29+22
Delicious	6+3	7+5	133+177	77+183	77+15
Mediamill	54+3	56+3	335+238	306+286	306+101
EUR-Lex	493+62	514+72	941+22,492	900+22,761	900+214
BioASQ	571+428	623+561	-	-	1,380+553
Wiki10	1,651+251	1,692+262	-	-	1,131+192
Delicious-200k	20,444+5,760	20,578+5,891	-	-	36,231+ 4,080
Amazon	14,931+984	15,044+1,194	-	-	38,510+ 221

Results vary greatly with respect to data sets too. For the small scale data sets, Subset LLDA achieves the best result in 7 out of 16 cases (four data sets times four evaluation measures), FastXML in 6 and PfastreXML in 3. For the large scale data sets, both FastXML and PfastreXML achieves the best result in 6 out of 16 cases, while Subset LLDA in 4. These small differences among the methods, suggest that there exists no one-size-fits-all algorithm for extreme learning tasks and each problem should be treated separately.

In Table 4, we additionally report the training and testing duration for each of the algorithms and data sets. The results show two clear tendencies, with Subset LLDA being significantly faster than the two other LLDA variants, while being slower across the majority of the data sets compared to FastXML and PfastreXML. We should note though, that our implementation could be further optimized by using much faster LDA implementations [3] and that both the extreme classification methods as well as our method can improve substantially by averaging over more trees or samples so a more detailed analysis should be conducted to assess the trade-off between duration and performance for each of the algorithms.

5 Conclusions and Future Work

In this work we have presented an extension of LLDA, to account for large-scale and XMLC tasks. Our algorithm Subset LLDA, proceeds in two stages, by first determining a set of relevant labels for a given test instance, and then constraining the CGS-LLDA algorithm to search only this label subspace for a solution. Experiments on eight data sets, with label sets sizes ranging from hundreds to hundreds of thousands, show a significant improvement over the best performing LLDA-based algorithms and competitive results with the state-of-the-art in extreme classification and suggest that Subset LLDA should be considered as a competitive alternative when dealing with XMLC tasks.

References

1. Bhatia, K., Jain, H., Kar, P., Varma, M., Jain, P.: Sparse local embeddings for extreme multi-label classification. In: *Advances in Neural Information Processing Systems*, pp. 730–738 (2015)
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. *J. Mach. Learn. Res.* **3**, 993–1022 (2003)
3. Chen, J., Li, K., Zhu, J., Chen, W.: WarpLDA: a cache efficient $o(1)$ algorithm for latent Dirichlet allocation. *Proc. VLDB Endowment* **9**(10), 744–755 (2016)
4. Griffiths, T.L., Steyvers, M.: Finding scientific topics. *Proc. Natl. Acad. Sci.* **101**(Suppl. 1), 5228–5235 (2004)
5. Jain, H., Prabhu, Y., Varma, M.: Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 935–944. ACM (2016)
6. Papanikolaou, Y., Foulds, J.R., Rubin, T.N., Tsoumakas, G.: Dense distributions from sparse samples: improved gibbs sampling parameter estimators for LDA. *J. Mach. Learn. Res.* **18**(62), 1–58 (2017)
7. Prabhu, Y., Varma, M.: FastXML: a fast, accurate and stable tree-classifier for extreme multi-label learning. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 263–272. ACM (2014)
8. Ramage, D., Hall, D., Nallapati, R., Manning, C.D.: Labeled LDA: a supervised topic model for credit attribution in multi-labeled corpora. In: *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009*, vol. 1, pp. 248–256. Association for Computational Linguistics, Stroudsburg (2009)
9. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. In: *Proceedings 20th European Conference on Machine Learning (ECML 2009)*, pp. 254–269 (2009)
10. Rubin, T.N., Chambers, A., Smyth, P., Steyvers, M.: Statistical topic models for multi-label document classification. *Mach. Learn.* **88**(1–2), 157–208 (2012)
11. Tsatsaronis, G., et al.: An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics* **16**, 138 (2015)
12. Tsoumakas, G., Katakis, I.: Multi-label classification: an overview. *Int. J. Data Warehouse. Min.* **3**(3), 1–13 (2007)

13. Weston, J., Makadia, A., Yee, H.: Label partitioning for sublinear ranking. In: Proceedings of the 30th International Conference on Machine Learning (ICML-13), pp. 181–189 (2013)
14. Yang, Y.: A study of thresholding strategies for text categorization. In: SIGIR 2001: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 137–145. ACM, New York (2001)
15. Yu, H.F., Jain, P., Kar, P., Dhillon, I.: Large-scale multi-label learning with missing labels. In: International Conference on Machine Learning, pp. 593–601 (2014)
16. Zhang, M.L., Zhou, Z.H.: A review on multi-label learning algorithms. *IEEE Trans. Knowl. Data Eng.* **99**(PrePrints), 1 (2013)



Third Party Data Clustering Over Encrypted Data Without Data Owner Participation: Introducing the Encrypted Distance Matrix

Nawal Almutairi^{1,2(✉)}, Frans Coenen^{1(✉)}, and Keith Dures^{1(✉)}

¹ Department of Computer Science, The University of Liverpool, Liverpool, UK

² Department of Information Technology, King Saud University,
Riyadh, Kingdom of Saudi Arabia
{n.m.almutairi,coenen,dures}@liverpool.ac.uk

Abstract. The increasing demand for Data Mining as a Service, using cloud storage, has raised data security concerns. Standard data encryption schemes are unsuitable because they do not support the mathematical operations that data mining requires. Homomorphic and Order Preserving Encryption provide a potential solution. Existing work, directed at data clustering, has demonstrated that using such schemes provides for secure data mining. However, to date, all proposed approaches have entailed some degree of data owner participation, in many cases the amount of participation is substantial. This paper proposes an approach to secure data clustering that does not require any data owner participation (once the data has been encrypted). The approach operates using the idea of an Encrypted Distance Matrix (EDM) which, for illustrative purposes, has been embedded in an approach to secure third-party data clustering - the Secure Nearest Neighbour Clustering (SNNC) approach, that uses order preserving and homomorphic encryption. Both the EDM concept and the SNNC approach are fully described.

Keywords: Privacy Preserving Data Mining
Secure Nearest Neighbour Clustering
Order Preserving Encryption · Homomorphic encryption

1 Introduction

Data Mining as a Service (DMaaS), using cloud storage, provides data owners with a set of useful tools for data analytics. Although cloud services provide a reliable infrastructure to host data and the potential for third party analytics, usage of such services entails issues of data confidentiality and security, and unauthorised data accesses (data leakage). Consequently, Privacy Preserving Data Mining (PPDM) approaches have been proposed to address these issues [1]. The typical PPDM approach is to provide the third party data miner with a version

of the data where sensitive data attributes have been either removed or modified using data transformation methods, such as data obfuscation, perturbation and anonymization. These methods tend to operate by introducing “statistical noise” to the sensitive attribute-values. This can then compromise the effectiveness of the data analysis; whilst, at the same time, it might still be possible to “reverse engineer” the original data values. Hence, data confidentiality cannot be guaranteed.

Data encryption can substantially guarantee data privacy and significantly mitigate against the risk of unauthorised data accesses. However, data encryption, in its standard form, prevents the application of any form of data mining, rendering it unsuited in the context of DMaaS. Data mining activities require data manipulation and data comparison of some form. A potential solution is Homomorphic Encryption (HE) [7], a form of encryption that supports a limited number of mathematical operations. HE schemes have been proposed that support addition, subtraction, multiplication and division of various forms. However, although the operations provided by HE schemes go some way to support DMaaS, they do not provide an entire solution; for example they do not support logical operations (over cypher-text) that data mining algorithms frequently require. One proposed solution [4,8] is to incorporate periodic recourse to data owners during the data mining process, so that the data owners can perform the data operations (on unencrypted data) that the adopted HE scheme does not support. For example, in the context of data clustering, the comparison of records. However, using this approach the amount of data owner participation is significant, calling in to question the advantages that DMaaS has to offer; although the approach does provide a suitable mechanism for collaborative secure data mining [11] and therefore does have merit. There has been some work that seeks to reduce the amount of data owner participation, of note is the idea of a 3-D Updateable Distance Matrix (UDM) presented in [2], but this still does not resolve the data owner participation issue. An alternative solution, in the context of collaborative data mining, is “secret sharing” [14], which aims to minimise data owner participation by introducing semi-honest and non-colluding third parties that decrypt, perform operations on “data pieces” (called shares) that hold no comprehensive information, and then re-encrypt, on behalf of the data owner. The global results can be reconstructed by knowing the individual results from several parties. Therefore, this again does not guarantee data confidentiality, whilst issues with data leakage remain.

From the foregoing, research directed at secure DMaaS has been predominantly focused on involving data owners, or constructing complex models to share secret keys, so as to resolve the current security issue associated with DMaaS. As noted above, these have significant limitations. Ideally, data owners should be able to package their data so that it is secure, send it to a third party for storage and analysis, and receive analysis results as and when required, without the need for any further communication whilst the analysis is taking place. In this paper a mechanism is proposed whereby secure third-party data mining (DMaaS) can be provided that does not entail any of the disadvantages of exist-

ing mechanisms. The fundamental idea, influenced in part by the UDM concept presented in [2], is to use a 2-D Encrypted Distance Matrix (EDM). The idea is illustrated in the context of Nearest Neighbour Clustering [3], we refer to this as the Secure Nearest Neighbour Clustering (SNNC) approach; however, the EDM idea clearly has wider application.

2 Previous Work

The main challenge of HE in the context of DMaaS in general, and data clustering in particular, is that HE schemes support only a limited number of arithmetic operations. As noted above, several theoretical and practical solutions to address this challenge have been proposed, these can be broadly categorised as either: (i) recourse to data owner when unsupported operations are required or (ii) utilising secret sharing techniques. Both have limitations in terms of communication complexity and security.

The key feature of the first category of solution is the realisation of data confidentiality by only permitting third party access to the HE data (without knowledge of the keys that have been used for the encryption). This means that data owner participation is required with respect to unsupported operations. The degree of participation depends on the nature of the mining to be undertaken. The worst case is when using what is known as Secure Multi-Party Computation (SMPC) [4, 11, 15], where the majority of data processing is conducted in-house by data owners. In the context of SMPC and k-Means clustering, as described in [11, 15], the third party data miner acts as a mediator and, on each iteration, calculates global cluster centroids; similarity measurement, assigning records to clusters and calculating local centroids are delegated to data owners. Data owners therefore do much of the work. In [4] k-Means clustering is also considered, but in this case, using an appropriate HE scheme; the third party data miner calculates distances between records and cluster centroids, and global centroids, whilst delegating similarity determination to data owners.

There has also been work directed at reducing the data owner's participation where the data owner provides static and dynamic *trapdoor* values to guide the third party data miner when comparing cypher-texts. One example is presented in [8] where K-Means clustering is used to illustrate the approach. However, data owner participation is still a significant requirement because the dynamic trapdoors need to be recalculated on each iteration of the K-Means clustering. The UDM concept proposed in [2] has the lowest data owner participation, illustrated in the context of k-Means clustering, where only very limited data owner participation is required on each iteration. However, the UDM (unlike the proposed EDM) is unencrypted; given that a UDM is essentially a set of linear equations this still presents a security threat. The idea of using user generated matrices holding data to support DMaaS has featured in other contexts. For example in [17] the data owner provides two matrices, of size $(2|A| + 2)$ and $(2|A| + 2) \times |A|$ (where $|A|$ is number of attributes), the two matrices are computed using a private matrix; the process is directed at supporting data classification.

However, data owner participation is still mandatory. In [16], an improvement of the scheme given in [17] was introduced that avoided data owner participation, however to do this part of the secret key is disclosed which in turn threatens data privacy.

The second category encompasses more recent work and features usage of some form of *secret sharing* scheme where at least two semi-honest, non-colluding, data miners perform computations by collaboratively decrypting private data on behalf of data owners, neither has access to the entire data set in unencrypted form. For example in [12, 14] a secret key is generated, by the collaborating parties, using a Threshold Paillier encryption scheme [5]. Secure computation protocols are used to allow the two parties to execute operations without requiring data owner participation. However, the collaborative nature of the computational protocols used induce communication overheads that make secret sharing very inefficient and thus not practical for large data sets. The requirement for at least two semi-honest and not-colluding data miners is also of concern.

The work presented in this paper does not fit neatly in either category, it does not require data owner participation and does not require secret sharing. In this context the proposed EDM mechanism is unique.

3 Preliminaries

Before considering the Encrypted Distance Matrix (EDM) concept and the Secure Nearest Neighbour Clustering (SNNC) approach in detail, the utilised encryption schemes are presented in this section.

3.1 Homomorphic Encryption: Liu's Scheme

Using the proposed SNNC approach, the raw data to be outsourced is encrypted using Liu's homomorphic encryption scheme as defined in [7]. The homomorphic properties of the scheme support the addition and subtraction of cypher-text, and the multiplication and division of cypher-text with real numbers. Although the proposed SNNC does not specifically utilise the homomorphic properties of Liu's scheme, the proposed solution is directed at providing a generic solution suited to many forms of secure data mining.

3.2 Order Preserving Encryption

A Distance Matrix holds the distances (differences) between each record in D with every other record in D . Using SNNC these distances are encrypted using an Order-Preserving Encryption (OPE) to give an Encrypted DM (EDM). Thus, the generated EDM holds the order of distance instead of real distance values.

The proposed OPE scheme is an amalgamation of two existing OPE schemes; [9, 10]. The key feature of the OPE is to obscure any data distribution

that might be included in the generated cypher-texts using the concept of “message space splitting” and “non-linear cypher space expansion”. The first step is to determine the “interval” of the message space $M = [l, r]$ and the “interval” of the cypher space $C = [l', r']$, where r is the maximum interval boundary and l is the minimum interval boundary, in such a way that $|C| \gg |M|$, as shown in Fig. 1. The next step is to randomly split the message space into successive intervals, as also shown in Fig. 1. The cypher space C is then split into the same number of intervals. However, the length of the cypher space intervals is determined by the density of the data in the corresponding message space intervals in such a way that message space intervals that have high data densities have large corresponding cypher space intervals. A “one-to-many” encryption function is then adopted. With the respect to the work presented in this paper the adopted function is shown in Algorithm 1. The encryption function commences by retrieving the interval ID number of value x by calling the *interval* function (line 2). The maximum and minimum interval boundary for the message space interval, holding x , and the corresponding cypher space interval are retrieved in lines 3 and 4. These values are used in lines 5 to 7 to generate the cypher x' . The δ_i variable in line 6 is a random number mapped from $[0, Sens \times Scale]$ where *Sens* is the minimum distance between the plain-text values in the data, as presented in [9], and scale value as calculated in line 5. This value guarantees that different cypher-texts, for the same plain-text value, will be generated on different occasions thus obscuring the data frequency.

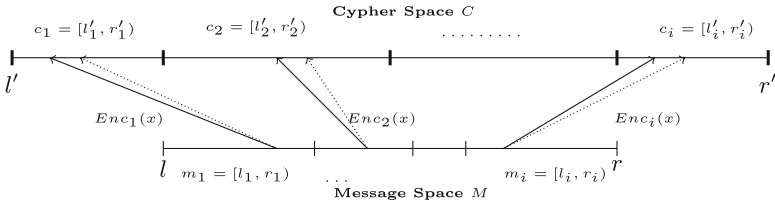


Fig. 1. Message and expanded cypher space splitting

Algorithm 1. Order Preserving Encryption algorithm

- 1: **procedure** ENC($x, Sens$)
 - 2: $i = \text{Interval}(x)$
 - 3: $[l_i, r_i] \leftarrow \text{Range}(i)$
 - 4: $[l'_i, r'_i] \leftarrow \text{Range}'(i)$
 - 5: $Scale_i = \frac{(l'_i - r'_i)}{(l_i - r_i)}$
 - 6: $\delta_i = \text{Random}(0, Sens \times Scale_i)$
 - 7: $x' = l'_i + Scale_i \times (x - l_i) + \delta_i$
 - 8: **Exit with** x'
-

4 Encrypted Distance Matrix (EDM) Generation

Regardless of whether standard or HE encryption is used, the encryption randomly translates the plain-text values in a given data set D to cypher-texts in such a way that any value ordering that existed in the original plain-text values is not preserved. Therefore, comparison operations cannot be directly applied and thus even the most trivial forms of data analysis cannot be performed. The idea presented in this paper is thus to use an Encrypted Distance Matrix (EDM), that holds encrypted distances between data values, so that comparisons can be conducted. An EDM is a 2D matrix where the first and second dimensions represent the records in D . The matrix is symmetric about the leading diagonal, thus only the leading triangle needs to be considered. EDM generation is done in two steps: (i) distance calculation and (ii) encryption. Given a data set $D = \{r_1, r_2, \dots, r_n\}$, where each record r_x is a feature vector comprised of a set of values $\{v_{x_1}, v_{x_2}, \dots, v_{x_a}\}$, a distance matrix, $DM(x, y)$, is calculated using:

$$DM(x, y) = \sum_{i=1}^{i=a} (v_{x_i} \sim v_{y_i}) \quad (1)$$

A DM calculated in this manner, as in the case of the UDM proposed in [2], essentially comprises a set of linear equations which might present a security threat. Therefore, the second step is to encrypt the data, but so that ordering is preserved. To this end the OPE scheme given in Subsect. 3.2 above was used.

5 Secure Nearest Neighbour Clustering

This section presents the proposed Secure Nearest Neighbour Clustering (SNNC) approach designed to operate over encrypted data and without any further user participation once the data has been outsourced. The clustering process, like the EDM generation process, has two steps: (i) data preparation conducted by the data owner and (ii) consequent clustering conducted by the third party. The first is discussed in Subsect. 5.1 below, and the second in Subsect. 5.2.

5.1 Data Owner Data Preparation

During the initial data preparation step the data owner pre-processes the data to be outsourced by replacing the categorical (or labelled) data with discrete integers values before any further processing. The processed data is then used to generate the required EDM after which the data is encrypted to give D' . Once the data has been successfully outsourced no further data owner participation will be required (other than receiving the final clustering result).

Algorithm 2. Secure Nearest Neighbour Clustering

```

1: procedure SECURENEARESTNEIGHBOURCLUSTERING( $D', EDM, \sigma'$ )
2:    $K_1 = \{r'_1\}$ 
3:    $K = \{K_1\}$ 
4:    $k = 1$ 
5:   for  $i = 2$  to  $i = |D'|$  do
6:     Find the  $r'_m$  in some cluster in  $K$  where the  $EDM[r'_i, r'_m]$  is the smallest
7:     if  $EDM[r'_i, r'_m] \leq \sigma'$  then
8:        $K_m = K_m \cup r'_i$ 
9:     else
10:       $k + +$ 
11:       $K_k = \{r'_i\}$ 
12:   Exit with  $K$ 

```

5.2 Third Party Clustering: SNNC Algorithm

The SNNC is conducted by the third party data miner following a process similar to that used for standard NNC [3]. The pseudo code presented in Algorithm 2 summarises this process. The input is the encrypted data set D' , the EDM (previously submitted to the third party) and the desired threshold σ' . The algorithm commences by assigning the first record r'_1 to the first cluster K_1 (lines 2 and 3). Next, the number of generated clusters so far is set to be 1 (line 4). A loop is then entered (lines 5 to 11) that iteratively clusters the remaining records in D' . A feature of the SNNC algorithm is that the threshold value σ is also encrypted, to give σ' , using the proposed OPE scheme so that the third party data miner processes the order of distances not real distance values. The record r'_i will be assigned to a cluster if there exists a record r'_m whose order of distance from r'_i is less than or equal to σ' using the EDM concept (lines 6 to 8). If no such record is found, a new cluster is created for r'_i (lines 10 and 11). The algorithm will continue until all records in D' are assigned to clusters and exits with a cluster configuration K .

6 Evaluation

The evaluation of the proposed SNNC approach is presented in this section. Extensive experiments were conducted to evaluate both the SNNC approach and the EDM concept. Fifteen data sets from the UCI data repository were used [6], these were selected so that data sets of a variety of sizes and different numbers of classes could be considered. The data sets are listed in Table 1. The implementation was done using the Java programming language. The evaluation criteria considered were: (i) data owner participation, (ii) clustering efficiency, (iii) comparative clustering accuracy and (iv) security. Each is considered in further detail in Subsects. 6.1, 6.2, 6.3 and 6.4.

6.1 Data Owner Data Preparation Run Time Complexity

Figure 2 (a to c) shows the runtime complexity recorded to encrypt the data, and calculate and encrypt the Distance Matrix (DM). The time to encrypt the data is correlated to the number of data records times the number of attributes in each data set. From the figure it can be seen that negligible time is required to encrypt the data, the recorded time to encrypt the largest data set (Arrhythmia) was 65 ms. In the case of calculating and encrypting the DM to produce the desired EDM, the runtimes were longer compared to data encryption although inspection of the figure shows that it is not significantly so. The reported time to calculate and encrypt the EDM for (Banknote authent.) was the highest; 876 ms to calculate the DM and 1509 ms to encrypt it. Additional records can be added, as and when they arrive, without necessitating re-encryption of the existing data. Once encrypted, the data and EDM are sent to the third party, no further data owner participation is required.

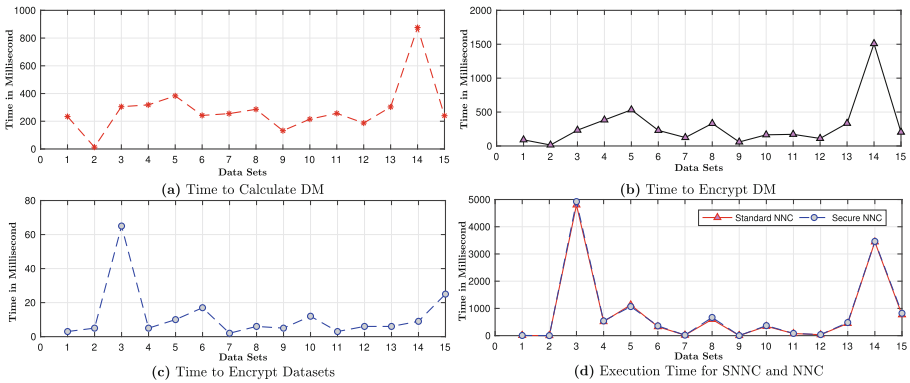


Fig. 2. Runtimes for data owner data preparation and NNC/SNNC execution

6.2 Clustering Efficiency

A comparison of the runtime required to cluster the data using standard NNC and SNNC is presented in Fig. 2(d). From the figure it can be seen that the difference in execution time is minimal. It can be concluded, at least in the context of NNC, that secure DMaaS using the proposed SNNC mechanism does not introduce a significant efficiency overhead (once the data has been encrypted).

6.3 Clustering Accuracy

In context of accuracy, cluster configuration “correctness” was measured by comparing the final clustering results obtained using standard NNC with those generated using the proposed SNNC approach; the SNNC approach should produce

Table 1. Cluster Configuration Comparison using Standard NNC and SNNC.

Data set	$R \times C$	Num. labels	σ	Standard NNC		Secure NNC	
				Num. cluster	Sil. coef.	Num. cluster	Sil. coef.
1. Iris	150×4	4	3.00	2	0.722	2	0.722
2. Lung cancer	32×56	3	0.10	32	1.000	32	1.000
3. Arrhythmia	452×279	16	1980.00	16	0.889	16	0.889
4. Blood transfusion	748×4	2	1046.00	4	0.895	4	0.895
5. Pima Ind. Diabetes	768×8	2	498.00	2	0.741	2	0.741
6. Chronic Kidney Dis.	400×24	2	952.00	16	0.981	16	0.981
7. Seeds	210×7	3	4.00	2	0.579	2	0.579
8. Breast Cancer	699×9	2	20.00	6	0.470	6	0.470
9. Breast Tissue	106×9	6	990.00	38	0.999	38	0.999
10. Dermatology	366×34	6	26.00	8	0.745	8	0.745
11. Ecoli	336×7	8	0.76	7	0.881	7	0.881
12. Parkinsons	195×22	2	91.00	8	0.930	8	0.930
13. Ind. Liver Patient	583×10	2	100.00	98	0.997	98	0.997
14. Banknote authent.	1372×4	2	11.00	16	0.752	16	0.752
15. Libras Movement	360×90	15	10.00	19	0.753	19	0.753

cluster configurations equivalent to those produced using standard NNC to prove that the proposed solution is operating correctly. This was measured by the Silhouette Coefficients (Sil. Coef.) [13]. Table 1 gives the results obtained in the context of both standard NNC and SNNC; columns 6 and 8. From the table it can be seen that the clustering configurations produced using SNNC were identical to those produced using standard NNC as evidenced by the Silhouette Coefficients obtained using the same σ threshold (shown in column 4).

6.4 Security Analysis

Security was evaluated in terms of the potential attacks that could be directed at the proposed secure clustering algorithm. The security of the proposed clustering approach relies on the security of: (i) Liu’s scheme for encrypting the raw data and (ii) the OPE used for encrypting the DM. Liu’s scheme is semantically secure as proven in [8], which means that adversaries cannot determine any information regarding the data from the cypher equivalents. In cryptography, when a scheme is said to be semantically secure this implies that the scheme is secure against Cypher-text Only Attacks (COAs). Therefore, with respect to the proposed secure clustering, adversaries that have access to the encrypted data set cannot readily threaten the system. In terms of the EDM, a COA could be used to extract statistical measures describing the frequency of distribution patterns which could be used to identify frequently occurring distributions which in turn could be used to identify the nature of plain-texts (if examples were available).

However the nature of the OPE scheme is such that the distribution is obscured using the concept of message space splitting and non-linear cypher space expansion. The one-to-many encryption function produces different cypher-texts for the same plain-text values thus obscuring the data frequency, especially when the scale intervals are large. Recall that data owner participation is avoided, thus Chosen Cypher-text attacks or Chosen Plain-texts attacks cannot be instigated with respect to the proposed secure clustering algorithm.

7 Conclusion

In this paper a mechanism for DMaaS has been proposed founded on the idea of an Encrypted Data Matrix (EDM). The approach was illustrated using a clustering scenario, the Secure Nearest Neighbour Cluster (SNNC) approach. The proposed method utilised Order Preserving Encryption (OPE) and Homomorphic Encryption (HE) to maintain data confidentiality. Unlike other proposed solutions to third party data clustering, the proposed approach does not require any data owner participation once the data (and the EDM) have been sent to the third party. The reported evaluation clearly demonstrates that the encryption schemes do not adversely affect the quality of data clustering. These are the same as when standard NNC is applied to the same data. For future work, the authors intend to investigate the utility of the EDM concept with respect to alternative clustering and classification algorithms.

References

1. Agrawal, R., Srikant, R.: Privacy-preserving data mining. *ACM Sigmod Rec.* **29**, 439–450 (2000)
2. Almutairi, N., Coenen, F., Dures, K.: K-means clustering using homomorphic encryption and an updatable distance matrix: secure third party data clustering with limited data owner interaction. In: Bellatreche, L., Chakravarthy, S. (eds.) *DaWaK 2017*. LNCS, vol. 10440, pp. 274–285. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-64283-3_20
3. Cover, T.M., Hart, P.E.: Nearest neighbour pattern classification. *IEEE Trans. Inf. Theory* **13**(1), 21–27 (1967)
4. Erkin, Z., Veugen, T., Toft, T., Lagendijk, R.L.: Privacy-preserving user clustering in a social network. In: 2009 First IEEE International Workshop on Information Forensics and Security (WIFS), pp. 96–100. IEEE (2009)
5. Hazay, C., Mikkelsen, G.L., Rabin, T., Toft, T.: Efficient RSA key generation and threshold paillier in the two-party setting. In: Dunkelman, O. (ed.) *CT-RSA 2012*. LNCS, vol. 7178, pp. 313–331. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27954-6_20
6. Lichman, M.: UCI machine learning repository (2013)
7. Liu, D.: Homomorphic encryption for database querying, 12 2013
8. Liu, D., Bertino, E., Yi, X.: Privacy of outsourced k-means clustering. In: *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, pp. 123–134. ACM (2014)

9. Liu, D., Wang, S.: Nonlinear order preserving index for encrypted database query in service cloud environments. *Concurr. Comput. Pract. Exp.* **25**(13), 1967–1984 (2013)
10. Liu, Z., Chen, X., Yang, J., Jia, C., You, I.: New order preserving encryption model for outsourced databases in cloud environments. *J. Netw. Comput. Appl.* **59**, 198–207 (2016)
11. Mittal, D., Kaur, D., Aggarwal, A.: Secure data mining in cloud using homomorphic encryption. In: 2014 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), pp. 1–7. IEEE (2014)
12. Rao, F.-Y., Samanthula, B.K., Bertino, E., Yi, X., Liu, D.: Privacy-preserving and outsourced multi-user k-means clustering. In: 2015 IEEE Conference on Collaboration and Internet Computing (CIC), pp. 80–89. IEEE (2015)
13. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* **20**, 53–65 (1987)
14. Samanthula, B.K., Elmehdwi, Y., Jiang, W.: K-nearest neighbor classification over semantically secure encrypted relational data. *IEEE Trans. Knowl. Data Eng.* **27**(5), 1261–1273 (2015)
15. Shen, Y., Han, J., Shan, H.: The research of privacy-preserving clustering algorithm. In: 2010 Third International Symposium on Intelligent Information Technology and Security Informatics, pp. 324–327 (2010)
16. Zhu, Y., Wang, Z., Zhang, Y.: Secure k -NN query on encrypted cloud data with limited key-disclosure and offline data owner. In: Bailey, J., Khan, L., Washio, T., Dobbie, G., Huang, J.Z., Wang, R. (eds.) PAKDD 2016, Part II. LNCS (LNAI), vol. 9652, pp. 401–414. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31750-2_32
17. Zhu, Y., Xu, R., Takagi, T.: Secure k-NN query on encrypted cloud database without key-sharing. *Int. J. Electron. Secur. Digit. Forensics* **5**(3–4), 201–217 (2013)

Pre-processing



An Efficient Prototype Selection Algorithm Based on Spatial Abstraction

Joel Luís Carbonera^(✉) and Mara Abel

Federal University of Rio Grande do Sul - UFRGS, Porto Alegre, Brazil
{j1carbonera,marabel}@inf.ufrgs.br

Abstract. Nowadays, machine learning and data mining approaches have been applied in huge volumes of data. In order to deal with this big data, techniques for prototype selection have been applied for reducing the data to a manageable volume and, consequently, for reducing the computational resources that are necessary to apply machine learning approaches. In this paper, we propose an efficient approach for prototype selection called PSSA. It adopts the notion of spatial partition for efficiently splitting the dataset in sets of similar instances. In a second step, the algorithm creates one prototype for each spatial partition and, finally, it selects the prototypes of the densest spatial partitions and that are far from other prototypes that were already selected. The approach was evaluated on 15 well-known datasets used in a classification task, and its performance was compared to those of 6 state-of-the-art algorithms, considering two measures: accuracy and reduction. All the obtained results show that, in general, the proposed approach provides a good trade-off between accuracy and reduction, with a significantly lower running time, when compared to other approaches.

Keywords: Prototype selection · Data reduction
Data mining · Machine learning · Big data

1 Introduction

In recent years, we have observed a dramatic increase in our capability to collect data. These resulting large collections of data, which is collectively called *big data*, challenge even the capabilities of our current data mining and machine learning approaches [10]. In this scenario, *prototype selection* has been viewed as an alternative to overcome the challenges raised by the size of these datasets.

Prototype selection is a data-mining (or machine learning) pre-processing task that consists in producing a smaller representative set of instances from the total available data, which can support the original task *with no performance loss* (or, at least, a reduced performance loss) [11]. Thus, every prototype selection strategy faces a *trade-off* between the *reduction rate* of the dataset and the resulting *performance quality* [9]. Also, it is important to emphasize that some works, such as [2], distinguish *prototype selection* from *instance selection*. While

instance selection involves a selection of a subset of actual instances of a dataset, *prototype selection* can replace the original instances by synthetic ones. In this work we adopt the expression *prototype selection* for generalizing both kinds of approaches.

Most of the proposed algorithms for prototype selection, such as [3, 15–19], are designed for preserving the *boundaries* between different classes in the dataset. This strategy acknowledges that border instances provide relevant information for supporting discrimination between classes [15].

In this paper, we propose an algorithm for prototype selection, called PSSA (Prototype Selection based on Spatial Abstraction). In an overview, the algorithm has three main steps: (I) it uses the notion of spatial partition for efficiently splitting the dataset in sets of instances that are similar to each other, then (II) it extracts prototypes from the sets of instances identified in the first step, and (III) it selects the prototypes that are in the densest partitions and that are far from other prototypes that were previously included in the resulting condensed set. Thus, in an overview, the algorithm creates a coarse representation of the dataset through a process of spatial abstraction, where a single prototype is generated from the set of instances of each spatial partition. And, in a second step, the algorithm selects a subset of the prototypes generated in the previous step. An interesting feature of the PSSA algorithm is that it allows the user to define the desired number of prototypes that will be extracted from the dataset. This level of control is not common in most of the approaches for prototype selection.

Our approach was evaluated on 15 well-known datasets and its performance was compared to the performance of 6 important algorithms provided by the literature, according to 2 different performance measures: accuracy and reduction. The accuracy was evaluated considering two classifiers: SVM and KNN. The results show that, when compared to the other algorithms, PSSA provides a good trade-off between accuracy and reduction, while presents a good running time.

Section 2 presents some related works. Section 3 presents the notation that will be used throughout the paper. Section 4 presents our approach. Section 5 discusses our experimental evaluation. Finally, Sect. 6 presents our main conclusions and final remarks.

2 Related Works

In this section, we discuss some important instance reduction methods. In this discussion, we consider T as the original set of instances in the training set and S , with $S \subseteq T$, as the reduced set of instances, resulting from the prototype selection process.

The *Condensed Nearest Neighbor* (CNN) algorithm [14] and *Reduced Nearest Neighbor* algorithm (RNN) [12] are some of the earliest proposals for prototype selection. Both can assign noisy instances to the final resulting set, are dependent on the order of the instances and have a high time complexity. The *Edited Nearest Neighbor* (ENN) algorithm [19] removes every instance that does not agree with

the label of the majority of its k nearest neighbors. This strategy is effective for removing noisy instances, but it does not reduce the dataset as much as other algorithms. In [18], the authors present 5 approaches, named the *Decremental Reduction Optimization Procedure* (DROP). These algorithms assume that those instances that have x as one of their k nearest neighbors are called the *associates* of x . Among the proposed algorithms, DROP3 has the best trade-off between the reduction of the dataset and the classification accuracy. It applies a noise-filter algorithm such as ENN. Then, it removes an instance x if its associates in the original training set can be correctly classified without x . The main drawback of DROP3 is its high time complexity. The *Iterative Case Filtering algorithm* (ICF) [3] is based on the notions of *Coverage set* and *Reachable set*. The coverage set of an instance x is the set of instances in T whose distance from x is less than the distance between x and its nearest enemy (instance with a different class). The Reachable set of an instance x , on the other hand, is the set of instances in T that have x in their respective coverage sets. In this method, a given instance x is removed from S if $|Reachable(x)| > |Coverage(x)|$. This algorithm also has a high running time. In [15], the authors adopted the notion of *local sets* for designing complementary methods for prototype selection. In this context, the local set of a given instance x is the set of instances contained in the largest hypersphere centered on x such that it does not contain instances from any other class. The first algorithm, called *Local Set-based Smoother* (LSSm) uses two notions for guiding the process: *usefulness* and *harmfulness*. The usefulness $u(x)$ of a given instance x is the number of instances having x among the members of their local sets, and the harmfulness $h(x)$ is the number of instances having x as the nearest enemy. For each instance x in T , the algorithm includes x in S if $u(x) \geq h(x)$. Since the goal of LSSm is to remove harmful instances, its reduction rate is lower than most of the instance selection algorithms. The author also proposed the *Local Set Border selector* (LSBo). Firstly, it uses LSSm to remove noise, and then, it computes the local set of every instance $\in T$. Then, the instances in T are sorted in the ascending order of the cardinality of their local sets. In the last step, LSBo verifies, for each instance $x \in T$ if any member of its local set is contained in S , thus ensuring the proper classification of x . If that is not the case, x is included in S to ensure its correct classification. The time complexity of the two approaches is $O(|T|^2)$. In [5], the authors proposed the *Local Density-based Instance Selection* (LDIS) algorithm. This algorithm selects the instances with the highest density in their neighborhoods. It provides a good balance between accuracy and reduction and is faster than the other algorithms discussed here. The literature provide some extensions to the basic LDIS algorithm, such as [4, 6].

Other approaches can be found in surveys such as [11, 13].

3 Notation

In this section, we introduce a notation adapted from [5] that will be used throughout the paper.

- $T = \{o_1, o_2, \dots, o_n\}$ is the non-empty set of n instances (or data objects), representing the original dataset to be reduced in the prototype selection process.
- $D = \{d_1, d_2, \dots, d_m\}$ is a set of m dimensions (that represent features, or attributes), where each $d_i \subseteq \mathbb{R}$.
- Each $o_i \in T$ is an m -tuple, such that $o_i = (o_{i1}, o_{i2}, \dots, o_{im})$, where o_{ij} represents the value of the j -th feature (or dimension) of the instance o_i , for $1 \leq j \leq m$.
- $val: T \times D \rightarrow \mathbb{R}$ is a function that maps a data object $o_i \in T$ and a dimension $d_j \in D$ to the value o_{ij} , which represents the value in the dimension d_j for the object o_i .
- $L = \{l_1, l_2, \dots, l_p\}$ is the set of p class labels that are used for classifying the instances in T , where each $l_i \in L$ represents a given class label.
- $l: T \rightarrow L$ is a function that maps a given instance $x_i \in T$ to its corresponding class label $l_j \in L$.
- $c: L \rightarrow 2^T$ is a function that maps a given class label $l_j \in L$ to a given set C , such that $C \subseteq T$, which represents the set of instances in T whose class is l_j . Notice that $T = \bigcup_{l \in L} c(l)$. In this notation, 2^T represents the *powerset* of T , that is, the set of all subsets of T , including the empty set and T itself.

4 The PSSA Algorithm

In this work, we assume that we can extract a prototype (a synthetic instance) from a given set of instances that are similar to each other, and that the resulting prototype is able to represent the information of the original set of instances. Considering this, it is possible to conceive a set of different strategies for prototype selection that follow the general schema represented in Algorithm 1, proposed in [7].

Algorithm 1. General schema for prototype selection

Input: A set instances T .

Output: A set P of prototypes that represent the information in T .

begin

$P \leftarrow \emptyset$;

foreach $l \in L$ **do**

$P^l \leftarrow$ a set $\{P_1^l, P_2^l, \dots, P_h^l\}$, such that $P_i^l \subseteq c(l)$ and P_i^l is a set of instances that are similar to each other;

foreach $P_i^l \in P^l$ **do**

$prot \leftarrow$ extraction of a prototype that abstracts the instances in P_i^l ;

$P \leftarrow P \cup \{prot\}$;

return P ;

Thus, in a first step, the general schema represented in Algorithm 1 basically defines, for each class label $l \in L$, a set P^l of sets of instances classified by l that

are similar to each other. In a second step, the algorithm extracts a prototype from each set $P_i^l \in P^l$. Notice that the sets in P^l can be generated by different kinds of approaches. One intuitive way of obtaining such sets is by applying clustering approaches, for example. Besides that, a prototype can be extracted from a set of instances by different approaches, including those used for generating centroids in clustering algorithms, such as *k-means* or *k-modes*.

In this paper, we propose the PSSA (*Prototype Selection based on Spatial Abstraction*) algorithm, which can be viewed as a specific variation of the general schema represented in Algorithm 1.

Before discussing the PSSA algorithm, it is important to consider the notion of *spatial partition*, which was also adopted in [7,8].

Definition 1. A *spatial partition* of a spatial region that contains a given set of objects $H \subseteq T$ is a set $\mathcal{SP}_H = \{s_1, s_2, \dots, s_m\}$, where:

$$\forall s_i \in \mathcal{SP}_H \rightarrow \exists d_i \in D \wedge s_i \subseteq d_i \tag{1}$$

$$\forall d_i \in D \rightarrow \exists s_i \in \mathcal{SP}_H \tag{2}$$

$$\begin{aligned} \forall s_i \in \mathcal{SP}_H \rightarrow s_i = [x, y] \wedge \\ x \geq \min(d_i, H) \wedge \\ y \leq \max(d_i, H) \end{aligned} \tag{3}$$

, considering $\min(d_i, H)$ as the lowest value of the dimension d_i within the set H , and $\max(d_i, H)$ as the greatest value of the dimension d_i within the set H .

That is, a *spatial partition* of the spatial region that contains a given set of objects H is intuitively a set of intervals, one for each dimension $d_i \in D$, defining a specific multidimensional region (a hyperrectangle) of the spatial region containing the objects in H . Figure 1 presents a dataset with 100 data objects in a 2D space with 12 spatial partitions.

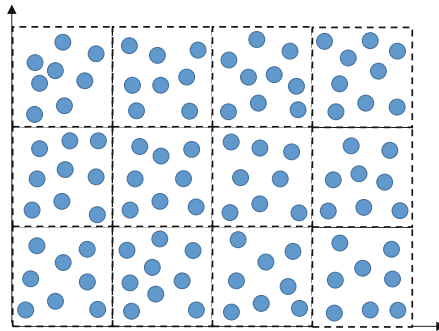


Fig. 1. Representation of a 2D space, with 12 *spatial partitions*.

Considering the notion of *spatial partition*, we can develop a specific version of the general schema provided by the Algorithm 1. In this version, firstly the algorithm identifies a set of spatial partitions for each class of objects of the dataset. In a second step, the algorithm extracts prototypes from the sets of instances located at the spatial partitions previously identified. At the final step, the algorithm selects the prototypes that are in dense regions and that are far from other prototypes previously included in the resulting set. The PSSA algorithm, formally represented in the Algorithm 2, adopts this general strategy.

The PSSA algorithm takes as input a set of data objects T , a value $n \in \mathbb{N}^*$, which determines the number of intervals in which each dimension of the dataset will be segmented; and a value $p \in [0, 1]$, which determines the expected number of prototypes that should be selected, as a percentage of the total number of instances ($|T|$) in the dataset. Considering this, the algorithm initializes P as an empty set and, for each $l \in L$, it:

1. Determines the number k of objects in $c(l)$ that should be included in P .
2. Determines the set R of sets of objects within $c(l)$, such that each set $r_i \in R$ is a set of objects contained within a specific *spatial partition* of the spatial region that contains the objects in $c(l)$. The set R is produced by the function *partitioning*, represented in Algorithm 3, which takes as input the set $c(l)$ of instances classified by l and the number n of intervals in which each dimension will be split.
3. Initializes three hash tables; *prot*, *density* and *selected*, which maps an integer value i (which is an index) to, respectively, the prototype of the i -th spatial region, a real value that represents the density of the i -th region and a boolean value that represents if the prototype of the i -th region was already included in the final resulting set.
4. For each i -th region (r_i) in R , the algorithm: extracts the prototype and includes it in *prot*[i], includes the density of the region ($|r_i|$) in *density*[i] and includes *FALSE* in *selected*[i] (because the prototype was not selected yet).
5. Normalizes the values in *density* by dividing them by the greatest value in *density*. Thus, after this step $\forall x \in \text{density} \rightarrow x \in [0, 1]$.
6. Defines *maxDist* as the distance between the two farthest prototypes in *prot*.
7. Finds the k prototypes that have a high density and that are far from the other prototypes already included in the final set. This is done by analysing every prototype in *prot* that were not selected yet and, in each iteration, selecting the candidate that has the greatest *weight*. Notice that the *weight* is the sum of the density of the region of the candidate prototype and the normalized distance (divided by *maxDist*) between the candidate and its nearest nearest prototype already included.

This strategy adopts two main assumptions. The first one is that prototypes extracted from dense regions are able to abstract more information of their neighbors than prototypes extracted from sparse regions. The second one is that, it is better to include prototypes that are far (and different) from each other, in order to keep a good representation of the variation of the instances of the class. These two assumptions are clearly present in the definition of the

weight of each candidate. Regarding this point, it is important to notice also that for calculating the *weight*, the algorithm sums up normalized versions of the considered *density* and *distance*. This is done for keeping this two values in the same scale (within the interval $[0, 1]$).

The function *partitioning*, on the other hand, is formalized by the Algorithm 3. This algorithm takes as input a set of instances H and a number $n \in \mathbb{N}^*$ of intervals in which each dimension will be segmented. It results in a set R of sets of instances, where each set $r_i \in R$ is the set of instances contained in some *spatial partition* of the spatial region that contains H . Initially, the algorithm defines R as an empty set. After, for each dimension $d_i \in D$, the algorithm defines $DRange$, which represents the range of the dimension d_i , as the absolute difference between the highest and the lowest value of d_i in H ; and $range_i$, which represents the range of an interval of d_i , as $\frac{DRange}{n}$. Notice that the intervals of each dimension are homogeneous. After, the algorithm considers *region* as a hash table whose keys are $|D|$ -tuples in the form of (x_1, x_2, \dots, x_j) , where each $x_i \leq n$ identifies one of the intervals of the i -th dimension in D . Also, for each key x , *region* stores a set objects, such that $region[x] \subseteq H$. Notice that a key of this hash table can be viewed as the identification of a given *spatial partition* and, therefore, $region[x]$ represents the set of objects located within the spatial partition identified by x . After, for each object $o \in H$, the algorithm:

1. Considers x as an empty $|D|$ -tuple.
2. Defines, for each dimension $d_i \in D$, the value of x_i , as the identification of the interval that contains the value $val(o, d_i)$, such that $x_i \leftarrow \lfloor \frac{val(o, d_i) - \min(d_i, H)}{range_i} \rfloor$. In this way, x determines the identification of the spatial partition that contains o .
3. Includes o as an element of $region[x]$.

After, for each key x of *region*, the algorithm includes the set $region[x]$ in R as an element. Thus, each element of R is a set of objects located in some spatial partition defined by the algorithm. Finally, the algorithm returns R .

Finally, the function *extractsPrototype*, adopted by the Algorithm 2 takes as input a set of instances $H \subseteq T$ and produces a $|D|$ -tuple c that represents the centroid of the instances in H . The function basically defines c as the centroid of H . Thus, it defines the value c_i of the i -th dimension of c as the average of the values of the dimension d_i within H .

It is important to notice that the Algorithm 3 uses the notion of spatial partition in an *implicit* way for identifying the set of instances that are included in each spatial partition. Also, it assumes that the spatial region that included all the elements in H is split in a set of non-overlapping spatial partitions, which covers all the set H .

Moreover, since the notion of *spatial partition* defined in this work can be applied only to datasets with quantitative (numerical) dimensions, the PSSA can be applied only to datasets with this kind of dimensions.

The algorithm uses an efficient algorithm for partitioning the whole data space of each class in a set of spatial partitions. This process is *linear* on the

Algorithm 2. PSSA algorithm

Input: A set instances T , a number $n \in \mathbb{N}^*$ of intervals, and a value $p \in [0, 1]$, which is the number of prototypes, as a percentage of $|T|$.

Output: A set *result* of prototypes.

begin

result $\leftarrow \emptyset$;

foreach $l \in L$ **do**

$k \leftarrow p \times |c(l)|$;

$R \leftarrow \text{partitioning}(c(l), n)$;

 Let *prot* be a hash table that maps an integer value i (index) to the prototype of the i -th spatial region;

 Let *density* be a hash table that maps an integer value i (index) to a real value that represents the density of the i -th region;

 Let *selected* be a hash table that maps an integer value i (index) to a boolean value that represents if the prototype of the i -th region was already included in the final resulting set;

foreach $r_i \in R$ **do**

$\text{prot}[i] \leftarrow \text{extractsPrototype}(r_i)$;

$\text{density}[i] \leftarrow |r_i|$;

$\text{selected}[i] \leftarrow \text{FALSE}$;

 Normalize the values in *density* by dividing them by the greatest value in *density*.

$\text{maxDist} \leftarrow$ the distance between the two farthest prototypes in *prot*;

while $j \leq |R|$ and $j \leq k$ **do**

$\text{maxWeight} \leftarrow 0$;

$\text{index} \leftarrow \text{null}$;

foreach i from 0 to $|R|$ **do**

if $\text{selected}[i] = \text{FALSE}$ **then**

$\text{minDist} \leftarrow \infty$;

foreach $p \in \text{result}$ **do**

$d \leftarrow \text{distance}(\text{prot}[i], p)$;

if $d < \text{minDist}$ **then**

$\text{minDist} \leftarrow d$;

$\text{weight} \leftarrow \text{density}[i] + \frac{\text{minDist}}{\text{maxDist}}$;

if $\text{weight} > \text{maxWeight}$ **then**

$\text{maxWeight} \leftarrow \text{weight}$;

$\text{index} \leftarrow i$;

$\text{result} \leftarrow \text{result} \cup \{\text{prot}[\text{index}]\}$;

$\text{selected}[\text{index}] \leftarrow \text{TRUE}$;

return *result*;

Algorithm 3. partitioning

Input: A set instances H and a number $n \in \mathbb{N}^*$ of intervals.
Output: A set R of sets of instances.
begin
 $R \leftarrow \emptyset$;
 Let $range_i$ be the range of an interval of the dimension $d_i \in D$;
 foreach $d_i \in D$ **do**
 $DRange \leftarrow abs(max(d_i, H) - min(d_i, H))$;
 $range_i \leftarrow \frac{DRange}{n}$;
 Let $region$ be a hash table whose keys are $|D|$ -tuples in the form of
 (x_1, x_2, \dots, x_j) , where each x_i identifies one of the intervals of the i -th
 dimension within D . Also, for each key x , $region$ stores a set of objects,
 such that $region[x] \subseteq H$.;
 foreach $o \in H$ **do**
 Let x be an empty $|D|$ -tuple.;
 foreach $d_i \in D$ **do**
 $x_i \leftarrow \lfloor \frac{val(o, d_i) - min(d_i, H)}{range_i} \rfloor$;
 $region[x] \leftarrow region[x] \cup \{o\}$;
 foreach key x of $region$ **do**
 R includes $region[x]$ as its element;
 return R ;

number of instances. However, for finding the k prototypes of the class l (represented by $k(l)$, which is equivalent to $|c(l)| * p$, where p is the percentage defined by the user), it is necessary to analyse each prototype (of the set $p(l)$ of prototypes extracted for the class l) and to compare it with the prototypes already included in the resulting set. Thus, the time complexity of this step is proportional to $O(\sum_{i=1}^{k(l)} (|p(l)| - i) * i)$. With this in mind, the time complexity of the whole algorithm is proportional to:

$$O(\sum_{l \in L} (|c(l)|) + \sum_{i=1}^{k(l)} ((|p(l)| - i) * i)) \tag{4}$$

Besides that, it is important to notice that the PSSA can be extended for performing an instance selection task (instead of prototype selection). In order to do this, for example, the extended algorithm could include a last step in which it selects the nearest instance of each selected prototype and use this instance (instead of the prototype) in the other operations.

5 Experiments

For evaluating our approach, we compared the PSSA algorithm in a classification task, with 6 important prototype selection algorithms provided by the literature: DROP3, ENN, ICF, LSBo, LSSm and LDIS. We considered 15 well-known

datasets with numerical dimensions: cardiocography, diabetes, E. Coli, glass, heart-statlog, ionosphere, iris, landsat, letter, optdigits, page-blocks, parkinson, segment, spambase and wine. All datasets were obtained from the UCI Machine Learning Repository¹. Table 1 presents the details of the datasets that were used.

Table 1. Details of the datasets used in the evaluation process.

Dataset	Instances	Attributes	Classes
Cardiocography	2126	21	10
Diabetes	768	9	2
E. Coli	336	8	8
Glass	214	10	7
Heart-statlog	270	14	2
Ionosphere	351	35	2
Iris	150	5	3
Landsat	4435	37	6
Letter	20000	17	26
Optdigits	11240	65	10
Page-blocks	5473	11	5
Parkinson	195	23	2
Spambase	9544	58	2
Segment	2310	20	7
Wine	178	14	3

We use two standard measures to evaluate the performance of the algorithms: *accuracy* and *reduction*. Following [5, 15], we assume:

$$accuracy = \frac{Success(Test)}{|Test|} \quad (5)$$

and

$$reduction = \frac{|T| - |S|}{|T|}, \quad (6)$$

where *Test* is a given set of instances that are selected for being tested in a classification task, *Success(Test)* is the number of instances in *Test* correctly classified in the classification task, and *|T|* and *S* are the number of instances in the original and in the reduced datasets, respectively.

For evaluating the classification *accuracy* of new instances in each respective dataset, we adopted a SVM and a KNN classifier. For the KNN classifier, we considered $k = 3$, as assumed in [5, 15]. For the SVM, following [1], we adopted

¹ <http://archive.ics.uci.edu/ml/>.

the implementation provided by Weka 3.8, with the standard parametrization ($c = 1.0$, $toleranceParameter = 0.001$, $epsilon = 1.0E - 12$, using a polynomial kernel and a multinomial logistic regression model with a ridge estimator as calibrator).

Besides that, following [5], the accuracy and reduction were evaluated in an n -fold cross-validation scheme, where $n = 10$. Thus, firstly a dataset is randomly partitioned in 10 equally sized subsamples. From these subsamples, a single subsample is selected as validation data (*Test*), and the union of the remaining 9 subsamples is considered the *initial training set (ITS)*. Next, a prototype selection algorithm is applied for reducing the *ITS*, producing the *reduced training set (RTS)*. At this point, we can measure the *reduction* of the dataset. Finally, the *RTS* is used as the training set for the classifier, which is used for classifying the instances in *Test*. At this point, we can measure the accuracy achieved by the classifier, using *RTS* as the training set. This process is repeated 10 times, with each subsample used once as *Test*. The 10 values of accuracy and reduction are averaged to produce, respectively, the *average accuracy (AA)* and *average reduction (AR)*. Tables 2, 3 and 4 report, respectively, for each combination of dataset and prototype selection algorithm: the resulting *AA* achieved by the SVM classifier, *AA* achieved by the KNN classifier, and the *AR*. The best results for each dataset is marked in bold typeface.

In all experiments, following [5], we adopted $k = 3$ for DROP3, ENN, ICF, and LDIS. For the PSSA algorithm, we adopted $n = 5$ and $p = 0.1$, since this parametrization provides a good balance between accuracy and reduction. Besides that, for the algorithms that use distance (dissimilarity) function, we adopted the standard euclidean distance:

$$d(x, y) = \sqrt{\sum_{j=1}^m (x_j - y_j)^2} \quad (7)$$

Tables 2 and 3 shows that LSSm achieves the highest *accuracy* in most of the datasets, for both classifiers. This is expected, since that LSSm was designed for removing noisy instances and does not provide high reduction rates. Besides that, for most of the datasets, the difference between the accuracy of PSSA and the accuracy achieved by the other algorithms is not large. The average accuracy achieved by PSSA is equivalent to the average accuracy of LDIS, and similar to the average accuracy of DROP3. In cases where the achieved accuracy is lower than the accuracy provided by other algorithms, this can be compensated by a higher reduction produced by PSSA and by a much lower running time. Table 4 shows that PSSA achieves the highest *reduction* in most of the datasets, and achieves also the highest average reduction rate. This table also shows that, in some datasets (such as parkinson and segment), the PSSA algorithm achieved a reduction rate that is significantly higher than the reduction achieved by other algorithms, with a similar accuracy. It is important to notice that the parameter p provides control over the desirable reduction rate of PSSA. This level of control is absent in other algorithms.

Table 2. Comparison of the *accuracy* achieved by the training set produced by each algorithm, for each dataset, adopting a SVM classifier.

Algorithm	DROP3	ENN	ICF	LSBO	LSSM	LDIS	PSSA	Average
Cardiotocography	0.64	0.67	0.64	0.62	0.67	0.62	0.60	0.64
Diabetes	0.75	0.77	0.76	0.75	0.77	0.75	0.74	0.76
E. Coli	0.81	0.82	0.78	0.74	0.83	0.77	0.81	0.80
Glass	0.47	0.49	0.49	0.42	0.55	0.50	0.50	0.49
Heart-statlog	0.81	0.83	0.79	0.81	0.84	0.81	0.79	0.81
Ionosphere	0.81	0.87	0.58	0.45	0.88	0.84	0.82	0.75
Iris	0.94	0.96	0.73	0.47	0.96	0.81	0.83	0.82
Landsat	0.86	0.87	0.85	0.85	0.87	0.84	0.84	0.85
Letter	0.80	0.84	0.75	0.73	0.84	0.75	0.74	0.78
Optdigits	0.98	0.98	0.97	0.98	0.99	0.96	0.97	0.98
Parkinsons	0.85	0.87	0.85	0.82	0.87	0.82	0.83	0.84
Segment	0.91	0.92	0.91	0.80	0.91	0.89	0.88	0.89
Spambase	0.90	0.90	0.90	0.90	0.90	0.89	0.89	0.90
Wine	0.93	0.95	0.94	0.96	0.97	0.94	0.93	0.95
Average	0.77	0.78	0.73	0.69	0.79	0.75	0.74	0.75

Table 3. Comparison of the *accuracy* achieved by the training set produced by each algorithm, for each dataset, adopting a KNN classifier.

Algorithm	DROP3	ENN	ICF	LSBO	LSSM	LDIS	PSSA	Average
Cardiotocography	0.63	0.64	0.57	0.55	0.67	0.54	0.51	0.59
Diabetes	0.72	0.72	0.72	0.73	0.72	0.68	0.65	0.70
E. Coli	0.84	0.84	0.79	0.79	0.86	0.82	0.83	0.83
Glass	0.63	0.63	0.64	0.54	0.71	0.62	0.58	0.62
Heart-statlog	0.67	0.64	0.63	0.66	0.66	0.67	0.62	0.65
Ionosphere	0.82	0.83	0.82	0.88	0.86	0.85	0.85	0.84
Iris	0.97	0.97	0.95	0.95	0.96	0.95	0.96	0.96
Landsat	0.88	0.90	0.83	0.86	0.90	0.87	0.87	0.87
Letter	0.88	0.92	0.80	0.73	0.93	0.79	0.73	0.83
Optdigits	0.97	0.98	0.91	0.91	0.98	0.95	0.93	0.95
Parkinsons	0.86	0.88	0.83	0.85	0.85	0.74	0.75	0.82
Segment	0.92	0.94	0.87	0.83	0.94	0.88	0.87	0.89
Spambase	0.79	0.81	0.79	0.81	0.82	0.75	0.76	0.79
Wine	0.69	0.66	0.66	0.74	0.71	0.69	0.62	0.68
Average	0.81	0.81	0.77	0.77	0.83	0.77	0.75	0.79

Table 4. Comparison of the *reduction* achieved by each algorithm, for each dataset.

Algorithm	DROP3	ENN	ICF	LSBO	LSSM	LDIS	PSSA	Average
Cardiotocography	0,70	0,32	0,71	0,69	0,14	0,86	0,90	0,62
Diabetes	0,77	0,31	0,85	0,76	0,13	0,90	0,90	0,66
E. Coli	0,72	0,17	0,87	0,83	0,09	0,92	0,90	0,64
Glass	0,75	0,35	0,69	0,70	0,13	0,90	0,90	0,63
Heart-statlog	0,74	0,35	0,78	0,67	0,15	0,93	0,90	0,65
Ionosphere	0,86	0,15	0,96	0,81	0,04	0,91	0,90	0,66
Iris	0,70	0,04	0,61	0,92	0,05	0,87	0,90	0,58
Landsat	0,72	0,10	0,91	0,88	0,05	0,92	0,90	0,64
Letter	0,68	0,05	0,80	0,84	0,04	0,82	0,90	0,59
Optdigits	0,72	0,01	0,93	0,92	0,02	0,92	0,90	0,63
Page-blocks	0,71	0,04	0,95	0,96	0,03	0,87	0,90	0,64
Parkinsons	0,72	0,15	0,80	0,87	0,11	0,83	0,90	0,63
Segment	0,68	0,05	0,79	0,90	0,05	0,83	0,90	0,60
Spambase	0,74	0,19	0,79	0,82	0,10	0,82	0,90	0,62
Wine	0,80	0,30	0,82	0,75	0,11	0,88	0,90	0,65
Average	0,73	0,17	0,82	0,82	0,08	0,88	0,90	0,58

We also carried out experiments for evaluating the impact of the parameters n and p in the performance of PSSA. The Table 5 represents the accuracy achieved by an SVM classifier (with the standard parametrization of Weka 3.8), as a function of the parameters n and p , with n assuming the values 2, 5, 10 and 20 and p assuming the values 0.05, 0.1 and 0.2. In this experiment, we also considered the 10-fold cross validation schema. Notice that the tables present the measures grouped primarily by the parameter n , and within each value of n , they present the results for each value of p .

The experiment show that with $n = 2$, the algorithm achieves the poorer performance, although, there is no significant differences in the average accuracy considering the other values of n . The small differences that we can identify in the results achieved with different values of n cannot be explained solely in terms of the change of n . This suggest that this parameter interacts with the structure of the dataset in a complex way. Further investigations should identify the properties and constraints regarding this interaction. On the other hand, as the value of p increases, considering a fixed value of n , the average accuracy increases, in general (the only exception is with $n = 2$). This is expected, since as p increases, the total number os prototypes selected by the algorithm also increases. Since each prototype abstracts the local information of its spatial partition, in most of the cases, increasing the value of p allows the resulting set of prototypes to capture more local information of the dataset. These additional

Table 5. Comparing the accuracy achieved by an SVM classifier trained with prototypes selected by PSSA with different values of n and p . The best values for each value of n in each algorithm are marked in bold typeface.

Algorithm	n=2			n=5			n=10			n=20			Average
	p=0.05	p=0.1	p=0.2	p=0.05	p=0.1	p=0.2	p=0.05	p=0.1	p=0.2	p=0.05	p=0.1	p=0.2	
Cardiotocography	0,58	0,60	0,59	0,58	0,60	0,64	0,58	0,60	0,62	0,59	0,61	0,62	0,60
Diabetes	0,75	0,70	0,72	0,71	0,74	0,76	0,72	0,75	0,76	0,70	0,73	0,74	0,73
E. Coli	0,75	0,69	0,68	0,78	0,81	0,79	0,73	0,76	0,79	0,65	0,74	0,80	0,75
Glass	0,53	0,51	0,48	0,51	0,50	0,51	0,47	0,47	0,50	0,53	0,51	0,50	0,50
Heart-statlog	0,80	0,81	0,82	0,72	0,79	0,83	0,77	0,78	0,81	0,76	0,78	0,81	0,79
Ionosphere	0,84	0,85	0,89	0,84	0,82	0,86	0,77	0,80	0,85	0,83	0,85	0,86	0,84
Iris	0,89	0,85	0,87	0,89	0,83	0,93	0,87	0,85	0,92	0,81	0,82	0,94	0,87
Landsat	0,82	0,83	0,84	0,77	0,84	0,85	0,83	0,84	0,85	0,84	0,85	0,86	0,84
Letter	0,67	0,70	0,67	0,68	0,74	0,78	0,64	0,72	0,78	0,65	0,72	0,78	0,71
Optdigits	0,95	0,96	0,98	0,95	0,97	0,98	0,95	0,97	0,97	0,95	0,97	0,98	0,96
Parkinsons	0,78	0,84	0,86	0,82	0,83	0,84	0,84	0,81	0,84	0,81	0,83	0,84	0,83
Segment	0,82	0,82	0,80	0,81	0,88	0,90	0,83	0,88	0,90	0,85	0,88	0,90	0,86
Spambase	0,76	0,70	0,73	0,87	0,89	0,88	0,88	0,89	0,89	0,87	0,88	0,90	0,84
Wine	0,93	0,95	0,99	0,91	0,93	0,96	0,93	0,94	0,96	0,89	0,93	0,96	0,94
Average	0,73	0,72	0,73	0,72	0,74	0,77	0,72	0,74	0,76	0,71	0,74	0,76	0,73

prototypes allows the classifier to use the additional information to make more fine-grained distinctions in the classification process.

We also carried out a comparison of the running times of the prototype selection algorithms considered in our experiments. In this comparison, we applied the 7 prototype selection algorithms to reduce the 3 biggest datasets considered in our tests: *letter*, *optdigits* and *spambase*. We adopted the same parametrizations that were adopted in the first experiment. We performed the experiments in an Intel® Core™ i5-5200U laptop with a 2.2 GHz CPU and 8 GB of RAM. The Fig. 2 shows that, considering these datasets, the PSSA algorithm achieves the lowest average running time, in comparison to the other algorithms. Notice that

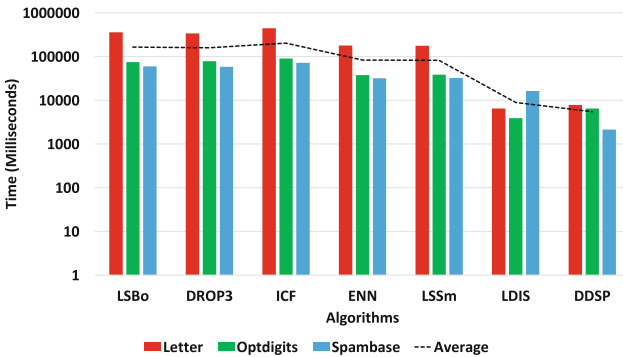


Fig. 2. Comparison of the running times of 7 prototype selection algorithms, considering the three biggest datasets. Notice that the time axis uses a logarithmic scale.

the Fig. 2 uses a *logarithmic scale* in the time axis, since there are big differences in the running time of the algorithms.

In summary, the experiments show that the PSSA algorithm has the lowest average running time, in comparison with other state-of-the-art algorithms. Also, although the running times of PSSA are similar to those of LDIS in some datasets, it is important to consider that DSSP has the advantage of allowing the user to select the desired number of prototypes. Besides that, PSSA also presents the highest reduction rates, while preserves a good accuracy, which is similar to the accuracy achieved by other algorithms, such as DROP3 and LDIS. These features suggest that PSSA is a promising algorithm for prototype selection in scenarios that a lower running time is critical.

6 Conclusion

In this paper, we proposed an efficient algorithm for prototype selection, called PSSA (Prototype Selection based on Spatial Abstraction). It adopts the notion of *spatial partition*, which, in an overview, is a set of intervals; one for each dimension of the dataset, which defines a multidimensional region in the dataset. The PSSA algorithm defines a set of spatial partitions for each class of objects in the dataset. After, it extracts a prototype of each spatial partition. And, in the last step, the algorithm selects the prototypes that are in the densest spatial regions previously identified and that are far from other prototypes that were previously identified. It is important to notice that the algorithm allows the user to specify the number of prototypes that should be selected. This provides a good level of control that is not a common feature in the prototype selection algorithms provided in the literature.

Our experiments show that PSSA provides the best reduction rates and a good balance between accuracy and reduction, when compared to other algorithms available in the literature. The empirical evaluation of running times showed that PSSA algorithm has an average running time that is lower than the running times of other state-of-the-art algorithms. These features make PSSA a promising algorithm for dealing with big volumes of data, in scenarios that the running time is critical.

In future works, we plan to investigate how to allow the algorithm to automatically identify the best way of splitting the dataset in a set of spatial partitions, without user intervention. Also, we plan to investigate the possibility of dealing with categorical dimensions in spatial partitions.

References

1. Anwar, I.M., Salama, K.M., Abdelbar, A.M.: Instance selection with ant colony optimization. *Procedia Comput. Sci.* **53**, 248–256 (2015)
2. Arnaiz-González, Á., Díez-Pastor, J.F., Rodríguez, J.J., García-Osorio, C.: Instance selection of linear complexity for big data. *Knowl. Based Syst.* **107**, 83–95 (2016)

3. Brighton, H., Mellish, C.: Advances in instance selection for instance-based learning algorithms. *Data Min. Knowl. Discov.* **6**(2), 153–172 (2002)
4. Carbonera, J.L.: An efficient approach for instance selection. In: Bellatreche, L., Chakravarthy, S. (eds.) *DaWaK 2017*. LNCS, vol. 10440, pp. 228–243. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-64283-3_17
5. Carbonera, J.L., Abel, M.: A density-based approach for instance selection. In: *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 768–774. IEEE (2015)
6. Carbonera, J.L., Abel, M.: A novel density-based approach for instance selection. In: *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 549–556. IEEE (2016)
7. Carbonera, J.L., Abel, M.: Efficient prototype selection supported by subspace partitions. In: *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE (2017)
8. Carbonera, J.L., Abel, M.: An efficient prototype selection algorithm based on dense spatial partitions. In: Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., Zurada, J.M. (eds.) *ICAISC 2018*. LNCS (LNAI), vol. 10842, pp. 288–300. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91262-2_26
9. Chou, C.H., Kuo, B.H., Chang, F.: The generalized condensed nearest neighbor rule as a data reduction method. In: *18th International Conference on Pattern Recognition, 2006, ICPR 2006*, vol. 2, pp. 556–559. IEEE (2006)
10. Fan, W., Bifet, A.: Mining big data: current status, and forecast to the future. *ACM SIGKDD Explor. Newsl.* **14**(2), 1–5 (2013)
11. García, S., Luengo, J., Herrera, F.: *Data Preprocessing in Data Mining*. Springer, Cham (2015). <https://doi.org/10.1007/978-3-319-10247-4>
12. Gates, G.W.: Reduced nearest neighbor rule. *IEEE Trans. Inf. Theory* **18**(3), 431–433 (1972)
13. Hamidzadeh, J., Monsefi, R., Yazdi, H.S.: IRAHC: Instance reduction algorithm using hyperrectangle clustering. *Pattern Recogn.* **48**(5), 1878–1889 (2015)
14. Hart, P.E.: The condensed nearest neighbor rule. *IEEE Trans. Inf. Theory* **14**, 515–516 (1968)
15. Leyva, E., González, A., Pérez, R.: Three new instance selection methods based on local sets: a comparative study with several approaches from a bi-objective perspective. *Pattern Recogn.* **48**(4), 1523–1537 (2015)
16. Lin, W.C., Tsai, C.F., Ke, S.W., Hung, C.W., Eberle, W.: Learning to detect representative data for large scale instance selection. *J. Syst. Softw.* **106**, 1–8 (2015)
17. Nikolaidis, K., Goulermas, J.Y., Wu, Q.: A class boundary preserving algorithm for data condensation. *Pattern Recogn.* **44**(3), 704–715 (2011)
18. Wilson, D.R., Martinez, T.R.: Reduction techniques for instance-based learning algorithms. *Mach. Learn.* **38**(3), 257–286 (2000)
19. Wilson, D.L.: Asymptotic properties of nearest neighbor rules using edited data. *IEEE Trans. Syst. Man Cybern. SMC* **2**(3), 408–421 (1972)



Web Usage Data Cleaning

A Rule-Based Approach for Weblog Data Cleaning

Amine Ganibardi^(✉)  and Chérif Arab Ali

University of Vincennes in Saint-Denis,
2 Rue de la Liberté, 93526 Saint-Denis, France
{aganibardi, aa}@ai.univ-paris8.fr

Abstract. This paper addresses the issue of Weblog Data cleaning within the scope of Web Usage Mining. Weblog data are information on end-user clicks and underlying user-agent hits recorded by web servers. Since Web Usage Mining is interested in end-user behavior, user-agent hits are referred to as noise to be cleaned before mining. The most referenced and implemented cleaning methods are the conventional and advanced cleaning. They are content-centric filtering heuristics, based on the requested resource attribute of the weblog database. These cleaning methods are limited in terms of relevancy, workability and cost constraints, within the context of dynamic and responsive web. In order to deal with dynamic and responsive web constraints, this contribution introduces a rule-based cleaning method focused on the logging structure rules. The rule-based cleaning method experimentation demonstrates significant advantages compared to the content-centric methods.

Keywords: Web Usage Mining · Web usage data preprocessing
Weblog data cleaning

1 Introduction

Web Usage Mining is the application of data mining techniques to usage data, termed weblog data, for different purposes, e.g., system enhancement and adaptation, personalization, recommendation and advertisement [1, 2]. Depending on the analysis purpose and context, the weblog data are completed with other data [3] and are mined using different mining techniques, e.g., statistics, classification, clustering, association/sequential rules and dependency modelling. Beyond the conventional steps of a mining process, Web Usage Mining preprocessing step tackles with three critical tasks, i.e., data cleaning, data structuration into single/unique users and visits, and episodes identification [1, 4–6]. The cleaning task references the relevance of the whole mining process [7–9]. It is meant to process the raw weblog data (R.WLDB) and provide a noise-free weblog database of clicks (C.WLDB). A R.WLDB refers to data recorded by web servers within Access Logfiles (ALF). It reflects usage activity on Website servers, i.e., end-user clicks and underlying user-agent hits [1, 8]. Notice that end-user clicks and user-agent hits relate to webpages URI (Uniform Resource Identifier) and their underlying displayed view components, respectively. Since Web Usage Mining is interested in end-user behavior,

user-agent hits are referred to as noise to be identified and removed before mining. Filtering hits from clicks is not trivial for two reasons, i.e., servers record requests interlaced in sequential order regardless of their source or type, websites resources may be set up as requestable interchangeably by end-users and user-agents [1, 8, 10].

On the basis of web usage mining reviewing papers from WEBKDD '99 up to 2017 [1–6, 11–13], the cleaning task can be mapped into three layers. The first is meant to identify clicks from hits. The second consists of selecting, from the identified clicks, those meaningful for the analysis purposes and context, e.g., the successful or unsuccessful requests, known single IP proxies and robots [3, 6, 7]. The third is performed downstream to data structuration and is rather a behavioral cleaning aimed at robot and outlier detection [3, 8, 14]. Thus, the core cleaning layer that tackles with generic noise is the identification of clicks from hits. In this regard, the most reported and implemented methods are the conventional and advanced cleaning. They are content-centric filtering heuristics based on the requested resource attribute of the R.WLDB. These cleaning methods are limited in terms of relevancy, workability and cost constraints, within the context of dynamic and responsive web content. In order to deal with dynamic and responsive web constraints, a rule-based cleaning method, focused on the logging structure rules, is introduced in the current paper. As the logging structure is insensitive to the logging content, the rule-based cleaning method experimentation demonstrates significant advantages compared to the conventional and advanced cleaning.

This paper is structured as follows: Sect. 2 depicts the cleaning methods and their limitations. Section 3 introduces the rule-based cleaning concept and method. Section 4 presents the experimental results analysis and evaluation.

2 Cleaning Methods Analysis

2.1 Data and Formalism

To better depict the cleaning method, the weblog data content, formats, attributes, and the underlying formalism are presented in the following. The most used ALF formats are the Common Log Format (CLF) and the Extended Common Log Format (ECLF) that contain information entries on usage activity [8]. The ALF entries content described in Table 1 represents the R.WLDB content. Given a R.WLDB of (n) requests (REQ) involving 7 entries instantiation recorded in an ALF as follows: $REQ_n = \{127.0.0.1, -, frank, 10/Oct/2000:13:55:36 -0700, GET/apache.gif HTTP/1.0, 200, 2326, http://apache.org/example.html, Mozilla/4.08 (Win*; ...; Brow*)\}$. Notice the following useful attributes and their abbreviation for the upcoming analysis: $[GET]$ the request method (REQ.MET), $[apache.gif]$ the requested resource (REQ.RES), $[apache]$ the requested resource name (REQ.RES.NAM), $[gif]$ the requested resource type (REQ.RES.TYP), $[apache.org]$ the referring resource root (REF.RES.ROO), $[example.html]$ the referring resource (REF.RES), $[example]$ the referring resource name (REF.RES.NAM), $[html]$ the referring resource type (REF.RES.TYP), $[Mozilla/4.08 (Win*; ...; Brow*)]$ the user-agent (USE.AGE). Each attribute occurrence is indexed $req_n(att_m)$. E.g., $req.res = apache.org$ in REQ_n above is indexed $req_n(req.res)$.

Table 1. Logfiles entries.

Entry	Description
IP address	IP address of the client (remote host) which made the request to the server
Client identity	RFC 1413 identity of the client determined by ident on the client's machine
Login	Login ID of the end-user requesting a resource (HTTP authentication)
Date and time	The time that the server finished processing the request
Request	The requested resource, request method, and http protocol version
Status code	Status code that the server sends back to the client (success, error, etc.)
Size	Indicates the size of the object returned to the client
Referrer	Indicates the resource that the client reports having been referred from
User-agent	Client browser information that it reports about itself (Browser and OS)

2.2 Related Methods

Conventional Cleaning. The conventional cleaning is based on the assumption that end-user clicks relate to html or equivalent web resource types as recommended by the World Wide Web Consortium (W3C) [1, 12]. All requests that relate to non-html resource types are assumed as user-agent requests and referred to as hits/noise. Thus, the cleaning attribute is the REQ.RES.TYP. In practice, undesired REQ.RES.TYP are set within a filtering knowledge database (FLT.KDB) that serves the cleaning heuristic. Thus, they are removed (FILTER-OUT) from the R.WLDB to come up with a cleaned weblog database (C.WLDB). The related cleaning process is given in Algorithm 1.

Algorithm 1. Conventional cleaning algorithm

```

01  INPUT DATA
02  R.WLDB
03  FLT.KDB
04  PROCESS
05  SCAN R.WLDB
06      IF reqn(req.res.typ) ∈ FLT.KDB FILTER-OUT reqn
07  OUTPUT DATA
08  C.WLDB

```

Advanced Cleaning. The advanced cleaning is based on prior knowledge on website URIs or a real-time extraction of those embedding clickable resources [10, 12]. The purpose is to set up a validation knowledge database (VLD.KDB) that contains all valid requestable resources imbedded in the website URIs and intended for end-user clicks. All requests that contain resources belonging to the VLD.KDB are referred to as clicks and are filtered into (FILTER-IN) the C.WLDB. Thus, the cleaning attribute is the REQ.RES. The related cleaning process is given in Algorithm 2.

Algorithm 2. Advanced cleaning algorithm

```

01  INPUT DATA
02  R.WLDB
03  VLD.KDB
04  PROCESS
05  SCAN R.WLDB
06  IF reqn(req.res) ∈ VLD.KDB FILTER-IN reqn
07  OUTPUT DATA
08  C.WLDB

```

2.3 Limitation Analysis

In order to achieve high quality outputs, the conventional and advanced cleaning need an exhaustive prior knowledge, extra-weblog and extra-cost factors in addition to the cleaning process cost. The conventional cleaning need exhaustive prior knowledge and extra-weblog data related to resource types set up as requestable by end-users or those intended for user-agents to be referred to as noise. Since there are no mandatory standards in this regard, the construction and updating of this prior FLT.KDB represent an additional cost factor besides the cleaning process cost. In addition, the conventional cleaning becomes obsolete in the case of websites based on frames without filetype extensions used as a cleaning attribute. The advanced cleaning is based on a VLD.KDB that consists of the REQ.RES embedded within the website structure. Thus, it needs extra-weblog data related to the website structure. In case of dynamic web design including automatic personalization and structure adaption, where the website content and structure are shifting continuously, the advanced cleaning becomes overlapping for servers and even impossible in case of personalized content [3]. This is due to the fact that the construction and updating of the VLD.KDB need a continuous extraction of the website URIs intended for end-users. Such a process is very complex to perform, represents an extra-cost factor, and may miss new/cancelled content if not synchronized with the cleaning process in real-time. Finally, the fact that the two methods need an exhaustive prior knowledge related to the website content, structure, and its resources intended for end-users, to set up the filtering heuristic, represents a serious weakness. Obtaining such an exhaustive prior knowledge within the analytics perspective of server owners is not obvious because server owners do not have necessarily this knowledge, unlike the website owners [15].

3 Rule-Based Cleaning Approach

3.1 Targeted Contribution Concept and Method

The targeted contribution is meant to provide a cleaning method that meets the advanced cleaning advantages (noise-free) without any need for extra-weblog data, prior knowledge, or extra-cost factors beyond the cleaning process. Such a method

aims to be workable within the context of dynamic and responsive Web in both of the analytics perspectives, i.e., server and website owners. This cleaning method filters the R.WLDB on the basis of the logging structure rules. Since the logging structure depends only on the logging format, the proposed method is insensitive to the repercussions of the dynamic web in terms of websites structure and content. Thus, the rule-based cleaning targets to overcome relevancy, workability and costs constraints of the advanced and conventional cleaning.

The method consists of three main steps, i.e., (1) the identification of logging structure features related to clicks request out of hits, (2) the abstraction of the features into cleaning rules, (3) the implementation of the rules within a cleaning algorithm to be experimented and evaluated. The rule-based cleaning is based on the rules related to the regular features of the clicks logging structure. Thus, a browsing simulation itemset drawn on a predefined relevant browsing items is performed through a dedicated software (Webserver Stress Tool 8.0.0.1010) on a mirrored website (Simple English Wikipedia) within a localhost server (Apache server) set to generate an ALF in the ECLF. The relevant browsing items concern the different manner of URIs browsing by an end-user, e.g., access by click on links from a search engine or a third-party website, typing the URI in the browser search bar, backward and bookmark navigation. The generated ALF is collected and the requests related to the browsed URIs (relevant item/end-user clicks) are highlighted within their underlying hits (irrelevant item/user-agent hits). The identified regular features in terms of the REQ.RES and REF.RES attributes of the clicks logging structure are abstracted to infer rules underlying to clicks out of hits. The inferred rules are combined into a filtering process that filters-in the relevant items into the C.WLDB. An implementation algorithm under R within Apache Spark API is set to perform a cleaning test on third-party websites ALF. The outcomes of the rule-based cleaning are compared to the conventional and advanced cleaning.

3.2 Logging Structure Features Identification (Step 1)

The attributes and functions of the identified regular features of the logging structure are exhibited in Table 2. Three generic features (a, b, c) are identified within the ALF structure: (a) Website access by URN (homepage/Domain Name/Uniform Resource Name) takes on an empty requested resource attribute, between the request and http protocol attributes. (b) Website browsing by URLs (content page) that is identified by the relationship between the requested resource/content page URL (Uniform Resource Locator) attribute and the referring resource attribute of its components. (c) Interfering noise with the regular features cited above that takes on responsive requests, namely, Style Queries (SQ) interlaced within the logged referring resource at the ALF referrer entry.

3.3 Cleaning Rule (Step 2)

The identified regular features can be formalized in three rules, i.e., URN access/clicks, URL access/clicks, and regular interfering noise/hits.

Table 2. Regular logging features.

Request	Requested Resource [name & type]	Referring Resource [name & type]	Activity
(a) HOMEPAGE ACCESS f (REQ.RES n)			
R n	Empty (URN)	Internal V External Referer/Page	Click
R $n+1$	Media 1	Home Page/URN	Hit
...	...	Home Page/URN	...
...	Media m	Home Page/URN	Hit
(b) CONTENT PAGE ACCESS f (REQ.RES n & REQ.RES $n+1$)			
...	Content Page (URL)	Internal V External Referer/Page	Click
...	Media 1	Content Page/URL	Hit
...	...	Content Page/URL	...
...	Media m	Content Page/URL	Hit
(c) INTERFERING NOISE f (REF.RES.TYP n) [style query type]			
...	Homepage/Content Page (URI)	Internal V External Referer/Page	Click
...	Media name	<i>Media style query</i>	Hit
...	...	Content Page/URL	Hit
R $n+x$	Media m	...	Hit

$$\text{URN ACCESS} \Rightarrow \text{req}_n((\text{req.met} \ \& \ \text{req.pro}) \neq \emptyset \ \& \ \text{req.res} = \emptyset) \quad (1)$$

$$\text{URL ACCESS} \Rightarrow \text{req}_n(\text{req.res}) = \text{req}_{n+1}(\text{ref.res}) \quad (2)$$

$$\text{INTERFERING NOISE} \Rightarrow \text{req}_n(\text{ref.res.typ}) \in \text{SQ} \quad (3)$$

The automatic inference and fill-in of the website DN within the empty attribute (req.res) bring the URN access rule to the URL access rule. The automatic inference of the DN given in (Eq. 4) is argued in the following. The referrer entry of an ALF takes on internal and external referrers (URIs). One click is referred by one external referrer and as many internal referrers as the number of the pointed page components. Since the internal referring resources (URIs) consist of the same root (Website DN), and the external referring ones relate to different roots, the most frequent root within the referrer entry is the Website DN. Thus, the website DN is the most frequent referring resources root.

$$\text{DN} = \arg. \max_{\text{freq}}(\text{REQ}(\text{ref.roo})) \quad (4)$$

The automatic inference of the style/media queries noise (SQ) given in (Eq. 5) is argued in the following. Considering that style queries are the unique page component of SQ type that may appear within the referrer entry, it is the least frequent resource type within the intersection of the requested and referring resource types.

$$SQ = \arg. \min_{\text{freq}}(\text{REQ.RES.TYP} \cap \text{REF.RES.TYP}) \quad (5)$$

Equations 4 and 5 represent statistical properties of the logging structure within the ECLF of an ALF. They have been tested and validated within the 6 ALFs that served in the experimentation section (Sect. 4). Thus, we make the assumption that they are generalizable within the ECLF of ALFs. Thus, after the automatic inference of the DN and the SQ, we end up with two rules only, i.e., access/clicks and noise rules.

$$\text{ACCESS} \Rightarrow \text{req}_n(\text{req.res}) = \text{req}_{n+1}(\text{ref.res}) \quad (6)$$

$$\text{NOISE} \Rightarrow \text{req}_n(\text{ref.res.typ}) \in \text{SQ} \quad (7)$$

3.4 Ruel-Based Cleaning Algorithm (Step 3)

Algorithm 3 draws the rule-based cleaning. The rule-based algorithm begins by parsing the weblog data to, first, infer the DN and fit it in the empty attributes where the requests method and http protocol are not missing. Then, it infers the SQ type that may appear in the referring entry. Once the SQ type is inferred, the requests whose referrers contain the SQ resource type are removed. Thus, the access rule can be applied to filter-in requests referred to as clicks into the C.WLDB.

Algorithm 3. Rule-based algorithm

```

01  INPUT DATA
02  R.WLDB
03  PROCESS
04  [ $\text{req}_n(\text{req.met} \ \& \ \text{req.pro} = \emptyset)$ ]  $\leftarrow \arg. \max_{\text{freq}} (\text{REQ}(\text{ref.roo}))$ ]
05   $\text{SQ} = \arg. \min_{\text{freq}} (\text{REQ.RES.TYP} \cap \text{REF.RES.TYP})$ 
06  WHILE  $\text{req}_n(\text{ref.res.typ} = \text{SQ})$  FILTER-OUT  $\text{req}_n$  WEND
07      IF  $\text{req}_n(\text{req.res}) = \text{req}_{n+1}(\text{ref.res})$  FILTER-IN  $\text{req}_n$ 
18  OUTPUT DATA
19  C.WLDB

```

4 Experimentation and Results Discussion

4.1 Experimental Data and Validation Reference

The rule-based cleaning algorithm as well as the conventional and advanced cleaning ones have been tested on a representative sample of 6 ALFs of third-party websites. The description of the experimental data is given in Table 3. This panel aims to be representative of the different practices in terms of web design, web content, and clicks/hits ratio. The involved ALFs relate to: a public administration (AL1.GOV), commercial websites (AL2.COM, AL3.COM, AL4.COM, AL5.COM), and our Laboratory website (AL6.LAB). The size of the ALFs is given in requests number.

They have been processed with the involved cleaning methods through their related Algorithms 1, 2 and 3, implemented under R within Apache Spark API.

Table 3. Experimental data description.

ALF	Website DN	ALF Source	Size	Clicks	Hits	Ratio
AL1.GOV	www.khanyounis.mun.ps	https://www.mosa.gov.ps/khanyounis.mun.ps/log/access.log	26 168	2 564	23 604	1/9
AL2.COM	almhuetter-raith.at	http://www.almhuetter-raith.at/apache-log/access.log	31 433	13 108	18 325	1/2
AL3.COM	www.facades.fr	http://igm.univ-mlv.fr/~cherrier/download/L1/access.log	5 945	987	4 958	1/5
AL4.COM	www.boring-log.com	https://www.scisoftware.com/boring-log.com/logs/apache-access.log	17 676	3 164	14 512	1/4
AL5.COM	www.megapeloteros.com	http://salablanda.com.ar/megapeloteros.com.access.log	680	48	632	1/13
AL6.LAB	www.ai.univ-paris8.fr/	Laboratory	145 227	25 391	119 836	1/6

The methods outputs are evaluated under the terms of a confusion matrix where: Size (SIZ) refers to the processed ALF size given in requests number; Positive (P) and Negative (N), respectively, indicate the number of clicks and hits the ALF contains. True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) indicate the valid and invalid requests, respectively, misreferred to as clicks or hits by the cleaning algorithm; False Positive Rate ($FPR = \frac{FP}{FP+TN}$) measures the noise rate; False Negative Rate ($FNR = \frac{FN}{FN+TP}$) measures the miss rate; Accuracy ($ACC = \frac{TP+TN}{P+N}$) measures the cleaning relevance. The validation reference (P and N number/labels) is composed of valid requestable resources by an end-user within the websites of the involved ALFs. This validation reference is set by a website checker software (Xenu's Link Sleuth 1.3.8) that reports, among others, the URIs embedding clickable resources intended for end-users.

4.2 Results Analysis

The cleaning results are depicted in Table 4. The analysis of the noise rate (FPR) of the CON.CLE algorithm shows that 53% consists of frames without filetype extensions in addition to unexpected responsive filetype extensions. 47% of the miss rate relate to different formats of clickable media content (non-html resources) that have been requested by end-users. The results in terms of FPR of the CON.CLE confirm the statement of its inappropriateness for frame-based web design. The results in term of FNR demonstrate the exhaustivity challenge in terms of prior FLT.KDB. Overall, the

Table 4. Cleaning results.

ALF	Size	P	N	TP	FP	TN	FN	FPR	FNR	ACC
<i>Advanced Cleaning (ADV.CLE)</i>										
AL1.GOV	26 168	2 564	23 604	2 564	0	23 604	0	0%	0%	100%
AL2.COM	31 433	13 108	18 325	13 108	0	18 325	0	0%	0%	100%
AL3.COM	5 945	987	4 958	987	0	4 958	0	0%	0%	100%
AL4.COM	17 676	3 164	14 512	3 164	0	14 512	0	0%	0%	100%
AL5.COM	680	48	632	48	0	632	0	0%	0%	100%
AL6.LAB	145 227	25 391	119 836	25 391	0	119 836	0	0%	0%	100%
SUM/AVG	227 129	45 262	181 867	45 262	0	181 867	0	0%	0%	100%
<i>Rule-based Cleaning (R.CLE)</i>										
AL1.GOV	26 168	2 564	23 604	2 564	0	23 604	0	0%	0%	100%
AL2.COM	31 433	13 108	18 325	13 108	0	18 325	0	0%	0%	100%
AL3.COM	5 945	987	4 958	987	0	4 958	0	0%	0%	100%
AL4.COM	17 676	3 164	14 512	3 164	0	14 512	0	0%	0%	100%
AL5.COM	680	48	632	48	0	632	0	0%	0%	100%
AL6.LAB	145 227	25 391	119 836	25 391	0	119 836	0	0%	0%	100%
SUM/AVG	227 129	45 262	181 867	45 262	0	181 867	0	0%	0%	100%
<i>Conventional Cleaning (CON.CLE)</i>										
AL1.GOV	26 168	2 564	23 604	2 252	312	22 947	657	1%	23%	96%
AL2.COM	31 433	13 108	18 325	12 725	383	6 474	11 851	6%	48%	61%
AL3.COM	5 945	987	4 958	795	192	4 852	106	4%	12%	95%
AL4.COM	17 676	3 164	14 512	3 040	124	11 715	2 797	1%	48%	83%
AL5.COM	680	48	632	43	5	627	5	1%	10%	99%
AL6.LAB	145 227	25 391	119 836	20 270	687	119 149	5121	1%	20%	96%
SUM/AVG	227 129	45 262	181 867	39 125	1 703	165 764	20 537	2%	27%	88%

CON.CLE is very noisy (ACC 88%). Both of the ADV.CLE and R.CLE tests give a noise-free (FPR) and a miss-free (FNR) rates. This is due to the fact that they are based on the same filtering attributes (REQ.RES) retrieved from two different sources. The R.CLE includes it in the access rule (Eq. 6) that is a function of the referring URIs within the ALF. However, the ADV.CLE retrieves it from the website URIs. Since the extracted website URIs contain all the possible requestable resources by click, the ADV.CLE provides noise-free outputs while the R.CLE reaches the same result in an optimized way, as it focuses only on the resource that has been requested by clicks identified by the access rule.

4.3 Results Evaluation

The CON.CLE is the most advantageous method in terms of relevancy balanced by workability and costs constraints. It provides high quality outputs with only one cost factor (the cleaning process). It does not need any extra-weblog data, making it insensitive to dynamic and responsive Web repercussions on the logging content. In addition, since it does not need any prior knowledge, it is workable for both website and server owner analytics perspectives. The ADV.CLE that gives also noise-free results is overlapping for

servers and proves to be unworkable in case of automatic and continuous shifting content and structure, specifically in case of personalized content. The CON.CLE is very noisy and proves to be obsolete in the case of responsive Web, based on frames. Both of the ADV.CLE and CON.CLE need exhaustive and prior knowledge on the involved website, making them limited to the website owner analytics perspective. Overall, the ADV.CLE and R.CLE can be combined where the early is optimized by the later, as they share a common filtering attribute from two different sources.

5 Conclusion

This contribution draws a critical analysis of weblog data cleaning. It depicts the related cleaning methods limitations in terms of relevancy, workability and cost constraints. The limitations of the analyzed methods are due to dynamic and responsive web repercussions on the logging content. These cleaning methods are content-centric and prove to be inappropriate to such a context, specifically within the analytics perspective of sever owners. Since the proposed method is focused on the logging structure, it demonstrates significant advantages compared to the content-centric cleaning methods in terms of relevancy balanced by workability and costs constraints. Finally, despite the limited size of the experimented ALF, the fact that the rule-based cleaning is based on the logging format, make it generalizable within the ECLF of ALFs, since the format does not depend on the logging size.

References

1. Srivastava, J., Cooley, R., Deshpande, M., Tan, P.-N.: Web usage mining: Discovery and applications of usage patterns from web data. *ACM SIGKDD Explor. Newsl.* **1**(2), 12–23 (2000)
2. Srivastava, M., Garg, R., Mishra, P.K.: Preprocessing techniques in web usage mining: a survey. *Int. J. Comput. Appl.* **97**(18), 1–9 (2014)
3. Kohavi, R.: Mining e-Commerce data: the good, the bad, and the ugly. In: Cheung, D., Williams, G.J., Li, Q. (eds.) *PAKDD 2001*. LNCS (LNAI), vol. 2035, p. 2. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45357-1_2
4. Facca, F.M., Lanzi, P.L.: Mining interesting knowledge from weblogs: a survey. *Data Knowl. Eng.* **53**(3), 225–241 (2005)
5. Langhnoja, S., Barot, M., Mehta, D.: Pre-processing: procedure on web log file for web usage mining. *Int. J. Emerg. Technol.* **2**(12), 5 (2012)
6. Chitraa, V., Thanamani, D.A.S.: Web log data cleaning for enhancing mining process. *Int. J. Commun. Comput. Technol.* **01**(03), 7 (2012)
7. Srivastava, J., Desikan, P., Kumar, V.: Web mining: Accomplishments and future directions. In: *National Science Foundation Workshop on Next Generation Data Mining (NGDM 2002)*, pp. 51–56 (2002)
8. Pabarskaite, Z., Raudys, A.: A process of knowledge discovery from web log data: systematization and critical review. *J. Intell. Inf. Syst.* **28**(1), 79–104 (2007)
9. Spiliopoulou, M., Mobasher, B., Berendt, B., Nakagawa, M.: A framework for the evaluation of session reconstruction heuristics in web-usage analysis. *Inform. J. Comput.* **15**(2), 171–190 (2003)

10. Pabarskaite, Z.: Implementing advanced cleaning and end-user interpretability technologies in web log mining. In: 2002 Proceedings of the 24th International Conference on Information Technology Interfaces, ITI 2002, pp. 109–113 (2002)
11. Dhandi, M., Chakrawarti, R.K.: A comprehensive study of web usage mining, pp. 1–5 (2016)
12. Srinivas, A.V.: A survey on preprocessing of web-log data in web usage mining. *Int. J. Modern Trends Sci. Technol.* **03**(02), 35–41 (2017)
13. Zhang, Q., Segall, R.S.: Web mining: a survey of current research, techniques, and software. *Int. J. Inf. Technol. Decis. Making* **7**(04), 683–720 (2008)
14. Spiliopoulou, M.: Web usage mining for Web site evaluation. *Commun. ACM* **43**(8), 127–134 (2000)
15. Zahran, D.I., Al-Nuaim, H.A., Rutter, M.J., Benyon, D.: A comparative approach to web evaluation and website evaluation methods. *Int. J. Pub. Inf. Syst.* **10**(1), 21–39 (2014)



Anonymization of Multiple and Personalized Sensitive Attributes

Jerry Chun-Wei Lin^{1,2(✉)}, Qiankun Liu¹, Philippe Fournier-Viger³,
Youcef Djenouri⁴, and Ji Zhang⁵

¹ School of Computer Science and Technology, Harbin Institute of Technology,
Shenzhen, China

mcqueenqkliu@gmail.com

² Department of Computing, Mathematics, and Physics,
Western Norway University of Applied Sciences (HVL), Bergen, Norway

jerrylin@ieee.org

³ School of Natural Sciences and Humanities, Harbin Institute of Technology,
Shenzhen, China

philfv8@yahoo.com

⁴ IMADA, Southern Denmark University, Odense, Denmark

djenouri@imada.sdu.dk

⁵ University of Southern Queensland, Queensland, Australia

Ji.Zhang@usq.edu.au

Abstract. In the past, many algorithms have presented to hide the sensitive information but most of them identify the sensitive information as the same for all users/transactions, which is not a situation happened in realistic applications. In this paper, we present the (k, p) -anonymity framework to hide not only the multiple sensitive information but also the personal sensitive ones. Extensive experiments indicated that the proposed algorithm outperforms the-state-of-the-art algorithms in terms of information loss and runtime.

Keywords: Anonymization · Cluster · Multiple sensitive information
Hierarchical attributes

1 Introduction

Data mining technique [6–8] is used to find the relationship among the items, which has widely applied in basket analytics for handling the transactional data [1, 3, 10]. Transactional data is a non-structural data, in which each transactional record is the set of events occurring in our daily life. The transactional data may record the purchase items (behaviors) of customers, medical diseases of patients, web query by end-users, etc. [26]. In specific applications such as medical records, each record may consist of sensitive information, such as personal identification, sexual orientation, medical diseases, etc. It may lead to the disclosure problem of privacy and cause the security threat if those information

is obtained by the malicious attackers. Besides, the attacker may infer the private and sensitive information if the non-sensitive information of each record is not the same.

The k -anonymity [20] is used to identify the $(k-1)$ transactions with the same attributes, in which the $(k-1)$ transactions are indistinguishable to each other. The l -diversity [18] was presented to ensure that the transactions with the same equivalence class are not less than l , thus the probability that the attacker to infer users' sensitive information is less than $1/l$. However, it does not successfully solve the attribute disclosure problem of k -anonymity. Besides, different users may have their defined sensitive information, but most existing algorithms set the sensitive information as the same importance for all users without personalized consideration.

Motivated by above problems, a novel and effective anonymity algorithm for transactional data is presented to take the individual factors of personal sensitivity into account, which can be used to solve the problems both in k -anonymity and l -diversity. A novel (k, p) -anonymity privacy-preserving framework is presented to solve the disclosure problem of attributes in k -anonymity and l -diversity. The proposed algorithm can also consider the personalized sensitive information, as well as the multiple sensitive information in the transactions for anonymization. From the conducted results, it showed that the proposed approach achieves better performance in terms of information loss and runtime than that of the state-of-the-art approaches.

2 Literature Review

Data mining techniques are powerful to find the relationship between the purchased items, but the confidential or private information may also be revealed, and cause the security threats [13–15]. The k -anonymity technology is used to protect personal privacy [20] against malicious attacks [21, 22] in relational database. This approach ensures that at least $(k-1)$ records become indistinguishable and the sensitive information can thus be hidden [17, 23, 25]. Poilis et al. [19] first considered the anonymization problem of relationship dataset. Wang et al. [24] proposed a novel utility metric and designed a K -anonymity framework for graph models. Doka et al. [4] defined the k -anonymity problem of maximal-utility and generalized it as a network flow problem. Most existing anonymity algorithms are, however difficult to directly apply to the transactional data since each transactional record is a non-structure data. Xu et al. [26] proposed a (h, k, p) -coherence concept for the anonymity problem of transactional data. Hsu et al. [11] proposed a k -anonymity algorithm for multi-patterns. Ghinita et al. [9] took the relevance of the purchase goods of customers into account to ensure that the sensitive items cannot be introduced by non-sensitive information, and solved the high dimension problem in the anonymization process for handling the transactional data. Lin et al. [16] presented an effective and efficient algorithm to achieve the anonymization for transactional data.

The above existing algorithms are based on k -anonymity concept, however, the leak of privacy since users can still be identified. Machanavajjhala et al. [18]

proposed two simple attack models, and pointed out that k -anonymity cannot solve those attacks. The l -diversity was presented to successfully solve these two attacks for anonymity. Li et al. [12] then developed a novel t -closeness privacy protection technique for anonymity. It requires, however, the distribution of sensitive attributes in any equivalence class to be close to that of the distribution of the attribute in the entire dataset.

3 Preliminaries and Problem Statement

A transactional dataset with n transactions is denoted as $D = \{T_1, \dots, T_n\}$, and each transaction in D is denoted as T_i . The items in D is denoted as $I = \{I_1, \dots, I_k\}$. Each T_i is a collection of a finite number of items in I . The personalized transactional dataset is shown in Table 1 by considering the personalized sensitive information, in which each item is correlated to a sensitivity of a user in the transaction.

Table 1. A personalized transactional dataset.

TID	Items with sensitivities
T_1	(1, 0.9); (3, 0.6); (4, 0.1); (5, 0.3); (7, 0.3); (11, 0.4)
T_2	(3, 0.3); (4, 0.2); (7, 0.7); (8, 0.2); (12, 0.7)
T_3	(3, 0.6); (4, 0.2); (8, 0.5); (9, 0.1)
T_4	(2, 0.3); (4, 0.2); (5, 0.6); (11, 0.8); (12, 0.9)
T_5	(2, 0.4); (5, 0.1); (7, 0.3); (10, 0.3); (12, 0.9)
T_6	(1, 0.1); (5, 0.4); (6, 0.5); (11, 0.8)
T_7	(2, 0.4); (4, 0.7); (5, 0.1); (7, 0.2); (11, 0.8)
T_8	(2, 0.2); (5, 0.2); (9, 0.5); (10, 0.1)
T_9	(1, 0.3); (3, 0.7); (4, 0.6); (6, 0.4); (8, 0.2); (12, 0.8)
T_{10}	(1, 0.2); (5, 0.2); (6, 0.3); (8, 0.5)
T_{11}	(3, 0.1); (5, 0.3); (12, 0.8)
T_{12}	(1, 0.2); (5, 0.3); (9, 0.4)

In Table 1, each item is represented as an integer with its correlated user's sensitivity. An item is considered as a sensitive item for further hiding if its sensitivity is no less than the given minimum sensitivity threshold. In this example, the minimum sensitivity threshold is assumed as 0.8. A hierarchical attribute of the items is also given as $\{A:(1, 7, 11), B:(2, 9), C:(3, 6), D:(4, 10), E:(5, 8), F:(12)\}$, which is used to map the original items into the representation of their hierarchical attributes. For example, the items 1, 7 and 11 can be generalized as the item (A). Thus, we can obtain a mapped dataset shown in Table 2. In Table 2, the attribute with sensitivity is against to the minimum sensitivity threshold for later anonymity. Each transaction consists of the hierarchical attributes, their

corresponding quantities, and the (multiple) sensitive information. In Table 2, each transaction contains the non-sensitive hierarchical attributes and sensitive items of the original database, in which each non-sensitive hierarchical attribute is corresponding to its occurrence frequency of the attribute in each transaction, and the sensitive information consists of the mapped attribute, original item, and the sensitivity.

Table 2. A mapped dataset under hierarchical attribute.

TID	Attributes with quantities	Attributes, items, sensitivities
T_1	A:2; C:1; D:1; E:1	A:1:0.9
T_2	A:1; C:1; D:1; E:1; F:1	-
T_3	B:1; C:1; D:1; E:1	-
T_4	B:1; D:1; E:1	A:11:0.8; F:12:0.9
T_5	A:1; B:1; D:1; E:1	F:12:0.9
T_6	A:2; C:1; E:1	A:11:0.8
T_7	A:1; B:1; D:1; E:1	A:11:0.8
T_8	B:2; D:1; E:1	-
T_9	A:1; C:2; D:1; E:1	F:12:0.8
T_{10}	A:1; C:1; E:2	-
T_{11}	C:1; E:1	F:12:0.8
T_{12}	A:1; B:1; E:1	-

Definition 1 ((k, p)-anonymity). *The (k, p)-anonymity framework satisfies that at least ($k-1$) transactions are indistinguishable in the anonymized database, and in each valid equivalence class, the sensitivity of the sensitive information is less than the given minimum sensitivity threshold p .*

Thus, the database is required to be modified to achieve (k, p)-anonymity. This process may cause the information loss, which is described as follows.

Definition 2 (Information loss, IL). *The information loss (IL) is the number of different items between the original dataset D and the anonymized dataset D'*

Problem Statement: The proposed (k, p)-anonymity framework sanitizes the database with ($k-1$) indistinguishable transactions under minimum sensitivity threshold p . Besides, the minimal IL is considered to be optimized between the original database and the sanitized one.

4 The Proposed (k, p)-anonymity Framework

In order to solve the above problems, an anonymous privacy protection algorithm is designed in this paper. The designed algorithm includes three stages such as:

(1) **data pre-processing**, (2) **clustering**, and (3) **anonymization**. In the data pre-processing stage, the mapped dataset is then transformed as the matrix representation, which can be easily to facilitate the clustering and anonymity operations. In the clustering stage, the non-sensitive parts are then clustered to be indistinguishably of user information. In the anonymization stage, the non-sensitive information is then checked against the cluster center to figure out the most closest transaction, then set it as the center of a equivalence class. The resting transactions in the same cluster are then sequentially processed according to their distance-ascending order to the center of the equivalence class. The condition of (k, p) -anonymity is then examined to decide whether this transaction can be assigned in the same equivalence class. This process can guarantee that the most similar transactions can be assigned in the same equivalence class to reduce the information loss (IL). Details of those three phases are described as below.

4.1 Data Pre-processing Phase

In order to facilitate higher similarity for anonymization, the proposed algorithm first performs the pre-processing procedure to transform the database as the matrix representation. The transformed results are shown in Table 3.

Table 3. Matrix representation of the mapped dataset.

TID	Non-sensitive items with quantities	Sensitive items
T_1	A:2, B:0, C:1, D:1, E:1, F:0	A:1, F:0
T_2	A:1, B:0, C:1, D:1, E:1, F:1	A:0, F:0
T_3	A:0, B:1, C:1, D:1, E:1, F:0	A:0, F:0
T_4	A:0, B:1, C:0, D:1, E:1, F:0	A:1, F:1
T_5	A:1, B:1, C:0, D:1, E:1, F:0	A:0, F:1
T_6	A:2, B:0, C:1, D:0, E:1, F:0	A:1, F:0
T_7	A:1, B:1, C:0, D:1, E:1, F:0	A:1, F:0
T_8	A:0, B:2, C:0, D:1, E:1, F:0	A:0, F:0
T_9	A:1, B:0, C:2, D:1, E:1, F:0	A:0, F:1
T_{10}	A:1, B:0, C:1, D:0, E:2, F:0	A:0, F:0
T_{11}	A:0, B:0, C:1, D:0, E:1, F:0	A:0, F:1
T_{12}	A:1, B:1, C:0, D:0, E:1, F:0	A:0, F:0

4.2 Clustering Phase

The purpose of anonymization algorithm is to ensure that the transactions become indistinguishable to each other, thus hiding the sensitive information

according to the user's privacy. The information loss (IL) is the major consideration to be achieved in the anonymization progress. Thus, the non-sensitive transactions with higher similarity are then assigned to the same cluster, which can be used to reduce the operation for modification in the original database. The pseudo-code of the designed clustering method is shown in Algorithm 1.

Algorithm 1. Proposed clustering algorithm

Input: A matrix of the mapped dataset.
Output: The set of the generated clusters, c_set .

```

1 set  $k$  as the number of clusters;
2 for each  $c_i \in c\_set$  do
3    $c_i.tids := null$ ;
4   for  $j \in matrix.row$  do
5      $c_i.center := random(matrix.row_j)$ ;
6 while termination is not satisfied do
7    $total\_dis := 0$ ; // total distances
8   for  $j = 0$  to  $matrix.row$  do
9      $c_i.center.min = argmin(distance(c_i.center.nos, matrix.row_j.nos))$ ;
10     $total\_dis+ = distance(c_i.center.min, matrix.row_j.nos)$ ;
11  for each  $c_i \in c\_set$  do
12     $c_i.center := \sum matrix.row_j / |c_i.tids|$ ;
13     $c_i.tids := null$ ;
14 return  $c\_set$ ;
```

In Algorithm 1, the k empty clusters is first given (Line 1). For each cluster, a transaction is randomly selected as the cluster center (Lines 4 to 5). The similarity of each transaction is then compared to the cluster center, and assign the transactions with high similarity into the same cluster. The distance of all transactions in a cluster is then calculated (Lines 7 to 10). After that, the new cluster center is generated by averaging the transactions within the same cluster, and the results are then iteratively performed for next generation (Lines 11 to 13). When the termination condition is achieved (while the sum of intra-cluster distance is no longer changed), the algorithm is then terminated. The results of Table 3 are then shown in Table 4.

Table 4. Generated clusters of the given example.

Cluster ID	TIDs	Cluster center
c_1	$T_1, T_2, T_6, T_9, T_{10}, T_{11}$	1 0 1 1 1 0
c_2	$T_3, T_4, T_5, T_7, T_8, T_{12}$	1 1 0 1 1 0

In Table 4, six transactions such as $T_1, T_2, T_6, T_9, T_{10}$ and T_{11} are grouped into the same cluster, and the other six transactions such as T_3, T_4, T_5, T_7, T_8 and T_{12} are then put into another cluster. By averaging each transaction within the same cluster, the cluster centers of two groups can be also calculated as 10110 and 110110, respectively.

4.3 Anonymization Phase

In order to avoid the security threats of k -anonymity and traditional l -diversity, it needs to consider the distribution of sensitive items in the process for generating the equivalence class. The developed anonymity algorithm is then shown in Algorithm 2.

Algorithm 2. Proposed anonymity algorithm

Input: c_set , the set of K cluster; L , anonymity degree; δ , minimum sensitivity threshold.

Output: EC_set , the set of equivalence class.

```

1 while  $|c\_set| \geq L$  do
2    $tid.close := \operatorname{argmin}\{\operatorname{distance}(c_i.center.nos, matrix.row_j.nos)\}$ ;
3    $EC_k.center.nos := \operatorname{normalization}(matrix.row_j.nos)$ ;
4    $EC_k.center.sen := \operatorname{normalization}(matrix.row_j.sen)$ ;
5   add  $tid.close$  to  $EC_k.tids$ ;
6   remove  $tid.close$  from  $c_j.tids$ ;
7   add  $EC_k$  into  $EC\_set$ ;
8   sort  $c.tids$  in distance-ascending order of
    $\operatorname{distance}(EC_k.center.nos, matrix.row_{tids}.nos)$ ;
9 for each  $tid \in c_j.tids$  do
10   $flag := \operatorname{judge}(matrix.row_{tids}.nos, \delta)$ ;
11  if  $flag == true$  then
12    add  $tid$  to  $EC_k.tids$ ;
13    remove  $tid$  from  $c_j.tids$ ;
14     $EC_k.center.sen += \operatorname{normalization}(matrix_{tids}.sen)$ ;
15  if  $|EC_k.tids| == L$  then
16    break;
17 for each  $tid \in c_j.tids$  do
18   $EC_k := (tid, \operatorname{argmin}\{\operatorname{distance}(EC_k.center.non, matrix.row_{tid}.nos)\})$ ;
19 return  $EC\_set$ ;
```

First, the most closest transaction of the non-sensitive part to the cluster center is then discovered (Line 2), and the it is thus set as the equivalence class (Line 3, Line 5). The sensitive part is also processed as the same way (Line 4). After that, this transaction is then removed from the cluster (Line 6). The remaining transactions in the cluster are then sorted in their *distance-ascending* order to the cluster center (Line 8). The most closest transaction to

the cluster center is then examined to decide whether it can be assigned to the equivalence class (Lines 10 to 16). If the corresponding value of a sensitive information is not less than 1, then its value is set as 1; otherwise, set it as 0. When the number of the remaining transactions in the cluster is less than k , the equivalence class will not be generated. The remaining transactions are then processed one by one to the nearest equivalence class. For each generated equivalent class, the algorithm generates new cluster center for the non-sensitive items according to the average value of the transactions within it. After the other transactions are assigned to the new cluster center of the non-sensitive part (Lines 17 to 18), the (k, p) -anonymity is thus achieved. The running example of the designed approach is shown in Table 5.

For example of cluster c_1 in Table 4, the distances of $T_1, T_2, T_6, T_9, T_{10}$, and T_{11} to the cluster center 101110 are respectively calculated as 1, 1, 2, 2, 2 and 2. The T_1 is then set as the cluster center of the equivalence class EC_1 ; T_1 is then removed from c_1 . The distances of the remaining five transactions to T_1 are respectively calculated as 2, 1, 2, 3, and 3; the transactions in c_1 are then re-sorted as T_6, T_2, T_9, T_{10} , and T_{11} according to their *distance*-ascending order to T_1 . Since T_6 and T_1 have the same sensitive hierarchical attribute of A , thus

Table 5. The equivalence class generated by the anonymity stage.

Equivalence class	TIDs	Non-sensitive cluster center
EC_1	T_1, T_2, T_9	1 0 1 1 1 0
EC_2	T_6, T_{10}, T_{11}	1 0 1 0 1 0
EC_3	T_3, T_4, T_8	0 1 0 1 1 0
EC_4	T_5, T_7, T_{12}	1 1 0 1 1 0

Table 6. The final anonymized results of the running example.

TID	Attributes with quantities	Attributes, items, sensitivities
T_1	$A:1; C:1; D:1; E:1$	$A:1:0.9$
T_2	$A:1; C:1; D:1; E:1$	-
T_3	$B:1; D:1; E:1$	-
T_4	$B:1; D:1; E:1$	$A:11:0.8; F:12:0.9$
T_5	$A:1; B:1; D:1; E:1$	$F:12:0.9$
T_6	$A:1; C:1; E:1$	$A:11:0.8$
T_7	$A:1; B:1; D:1; E:1$	$A:11:0.8$
T_8	$B:1; D:1; E:1$	-
T_9	$A:1; C:1; D:1; E:1$	$F:12:0.8$
T_{10}	$A:1; C:1; E:1$	-
T_{11}	$A:1; C:1; E:1$	$F:12:0.8$
T_{12}	$A:1; B:1; D:1; E:1$	-

if T_6 is put into EC_1 , the probability of the sensitive attribute A in 3-diversity equivalence class is calculated as 66%, which exceeds the given minimum sensitivity threshold as 60%; T_6 will not be assigned to EC_1 . After that, T_2 and T_6 will be grouped in EC_1 , and new cluster center of the non-sensitive data are generated by averaging the value of the transactions. The final results of the anonymized dataset is shown in Table 6, and each equivalence class satisfies (3, 60%)-anonymity condition.

5 Experimental Evaluation

Extensive experiments are then executed to verify the performance of the proposed algorithm compared to the traditional Gray-TSP [27] and the PTA [16]. It is also to notice that no existing works concerned the personalized sensitive information for anonymity. three real-life datasets [5] and a synthetic dataset [2] were used in the experiments to evaluate the performance of the compared algorithms. Parameters and characteristics of the datasets were respectively shown in Tables 7 and 8.

Table 7. Parameters of used datasets.

$\# D $	Total number of transactions
$\# I $	Number of distinct items
AvgLen	Average length of transactions
MaxLen	Maximal length of transactions

Table 8. Characteristics of used datasets.

Dataset	$\# D $	$\# I $	AvgLen	MaxLen
mushroom	8,124	119	23	Dense
pumsb	49,046	2113	74	Dense
accidents	340,183	468	33.8	Dense
T10I4D100K	100,000	870	10.1	Sparse

5.1 Information Loss

The IL ratio is used to evaluate the item differences between the original dataset and the anonymized one. The IL of three algorithms under varied k values with a fixed cluster number is then compared and shown in Fig. 1(a). Since the Gray-TSP and the PTA algorithms have not adopted the clustering method, thus the number of segments used in those two algorithms is used to instead of k

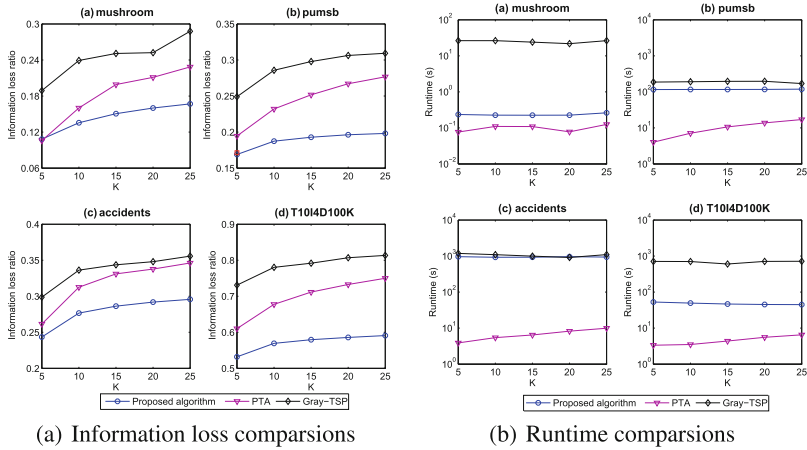


Fig. 1. Experiments under varied k .

value. The cluster numbers of the conducted dataset are respectively set as 150, 800, 6,000, and 1,000 for mushroom, pumsb, accidents, and T10I4D100K. The minimum sensitivity threshold is set as 75% for all datasets.

From Fig. 1(a), it can be observed that the IL of four algorithms increases along with the increasing of k values. The proposed algorithm obviously outperforms the others since it produces less IL .

5.2 Runtime

The runtime of the developed algorithm includes the pre-processing time, clustering time, and anonymization time. Similarly, the Gray-TSP and the PTA do not employ the clustering technique, thus the number of segment is used instead of the number of clusters used in the designed algorithms. The results with a fixed cluster number is then compared and shown in Fig. 1(b).

From Fig. 1(b), it can be seen that the proposed algorithm always requires less runtime than that of the Gray-TSP. Since the number of segments is fixed set, the runtime of the PTA is stable and its runtime increases along with the increasing of k value. Besides, it only concerns the search process of the shortest path and anonymization process, the runtime of PTA only slightly outperforms the designed algorithms in general cases. However, the information loss ratio of the PTA is larger than that of the designed algorithm. We then can conclude that the developed algorithm outperforms the PTA and the Gray-TSP, generally.

6 Conclusions

In this paper, we presented the (k, p) -anonymity framework to sanitize the customized and multiple sensitive information. Extensive experiments under

different parameters showed that the proposed algorithm cannot only hide the personalized sensitive information but also handle the multiple sensitive information for anonymization. Thus, the proposed (k, p) -anonymity solves the disclosure problem existing in the traditional k -anonymity and l -diversity technologies.

Acknowledgment. This research was partially supported by the Shenzhen Technical Project under JCYJ20170307151733005 and KQJSCX20170726103424709.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: The International Conference on Very Large Data Bases, pp. 487–499 (1994)
2. Agrawal, R., Srikant, R.: Quest synthetic data generator (1994). <http://www.Almaden.ibm.com/cs/quest/syndata.html>
3. Chen, M.S., Park, J.S., Yu, P.S.: Efficient data mining for path traversal patterns. *IEEE Trans. Knowl. Data Eng.* **10**(2), 209–221 (1998)
4. Doka, K., Xue, M., Tsoumakos, D., Karras, P.: k -anonymization by freeform generalization. In: ACM Symposium on Information, Computer and Communications Security, pp. 519–530 (2015)
5. Fournier-Viger, P., et al.: The SPMF open-source data mining library version 2. In: Berendt, B., et al. (eds.) ECML PKDD 2016. LNCS (LNAI), vol. 9853, pp. 36–40. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46131-1_8
6. Fournier-Viger, P., Lin, J.C.W., Kiran, R.U., Koh, Y.S., Thomas, R.: A survey of sequential pattern mining. *Data Sci. Pattern Recogn.* **1**(1), 54–77 (2017)
7. Gan, W., Lin, J.C.W., Fournier-Viger, P., Chao, H.C., Tseng, V.S., Yu, P.S.: A survey of utility-oriented pattern mining, [arXiv:1805.10511](https://arxiv.org/abs/1805.10511) (2018)
8. Gan, W., Lin, J.C.W., Fournier-Viger, P., Chao, H.C., Yu, P.S.: A survey of parallel sequential pattern mining, [arXiv:1805.10515](https://arxiv.org/abs/1805.10515) (2018)
9. Ghinita, G., Kalnis, P., Tao, Y.: Anonymous publication of sensitive transactional data. *IEEE Trans. Knowl. Data Eng.* **23**(2), 161–174 (2011)
10. Han, J., Pei, J., Yin, Y., Mao, R.: Mining frequent patterns without candidate generation: a frequent-pattern tree approach. *Data Min. Knowl. Discov.* **8**(1), 53–87 (2004)
11. Hsu, C.H., Tsai, H.P.: KAMP: preserving k -anonymity for combinations of patterns. In: IEEE International Conference on Mobile Data Management, pp. 97–102 (2013)
12. Li, N., Li, T., Venkatasubramanian, S.: t -closeness: privacy beyond k -anonymity and l -diversity. In: IEEE International Conference on Data Engineering, pp. 106–115 (2007)
13. Lin, C.W., Hong, T.P., Hsu, H.C.: Reducing side effects of hiding sensitive itemsets in privacy preserving data mining. *Sci. World J.* **2014**, Article ID 235837, 12 (2014)
14. Lin, C.W., Zhang, B., Yang, K.T., Hong, T.P.: Efficiently hiding sensitive itemsets with transaction deletion based on genetic algorithms. *Sci. World J.* **2014**, Article ID 398269, 13 (2014)
15. Lin, C.W., Hong, T.P., Yang, K.T., Wang, S.L.: The GA-based algorithms for optimizing hiding sensitive itemsets through transaction deletion. *Appl. Intell.* **42**(2), 210–230 (2015)

16. Lin, C.W., Liu, Q., Fournier-Viger, P., Hong, T.P.: PTA: an efficient system for anonymizing transaction databases. *IEEE Access* **4**, 6467–6479 (2016)
17. Kisilevich, S., Rokach, L., Elovici, Y., Shapira, B.: Efficient multidimensional suppression for K -anonymity. *IEEE Trans. Knowl. Data Eng.* **22**, 334–347 (2010)
18. Machanavajjhala, A., Kifer, D., Gehrke, J.: L -diversity: privacy beyond K -anonymity. *ACM Trans. Knowl. Discov. Data* **1**(1), 1–52 (2006)
19. Poulis, G., Loukides, G., Skiadopoulos, S.: Privacy-preserving anonymization of set-valued data. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 353–369 (2013)
20. Samarati, P., Sweeney, L.: Generalizing data to provide anonymity when disclosing information. In: *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, p. 188 (1998)
21. Sweeney, L.: k -anonymity: a model for protecting privacy. *IEEE Sec. Priv. Mag.* **10**(5), 557–570 (2012)
22. Sweeney, L.: Achieving k -anonymity privacy protection using generalization and suppression. *Int. J. Uncertain. Fuzziness Knowl. Based Syst.* **10**, 571–588 (2012)
23. Terrovitis, M., Mamoulis, N., Kalnis, P.: Privacy-preserving anonymization of set-valued data. *VLDB Endow.* **1**, 115–125 (2008)
24. Wang, Y., Xie, L., Zheng, B., Lee, K.C.K.: High utility K -anonymization for social network publishing. *Knowl. Inf. Syst.* **41**, 697–725 (2014)
25. Xu, J., Wang, W., Pei, J., Wang, X., Shi, B., Fu, W.C.: Utility-based anonymization using local recoding. In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–790 (2006)
26. Xu, Y., Wang, K., Fu, W.C., Yu, P.S.: Anonymizing transaction databases for publication. In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 767–775 (2008)
27. Xue, M., Karras, P., Raïssi, C., Vaidya, J., Tan, K.L.: Anonymizing set-valued data by nonreciprocal recoding. In: *ACM Knowledge Discovery and Data Mining*, pp. 1050–1058 (2012)



TRANS-AM: Discovery Method of Optimal Input Vectors Corresponding to Objective Variables

Hiroaki Tanaka^{1(✉)}, Yu Suzuki^{1,2(✉)}, Koichiro Yoshino^{1,3(✉)},
and Satoshi Nakamura^{1,2(✉)}

¹ Graduate School of Information Science, Nara Institute of Science and Technology,
Ikoma, Japan

{tanaka.hiroaki.sy2,ysuzuki,koichiro,s-nakamura}@is.naist.jp

² Data Science Centre, Nara Institute of Science and Technology, Ikoma, Japan

³ PRESTO: Fundamental Information Technologies toward Innovative Social System
Design. Japan Science and Technology Agency, Kyoto, Japan

Abstract. In various fields, ensemble models by supervised learning are effective, but the models cannot tell us how to modify the input vector so that we will increase the objective variable more than a given threshold or decrease it less than the threshold. In this paper, we propose a method, TRANS-AM, that can discover an input vector satisfying the condition of changing of the objective variable in regression problems by using a property of regression tree. The regression tree splits input space into subspaces. There are subspaces with corresponding objective variables satisfying such a condition. By transforming the input vector to new input vectors belonging to one of the subspaces, we can discover a new input vector whose distance from the original input vector is minimum by satisfying the condition to change the objective variable. The reason for “minimum” is the cost—if the new input vector is far from the original one, we need the significant cost to modify the original input vector to the new one. We evaluated the proposed method through numerical simulations and investigated that the proposed method works well; the ratio of the number of discovered input vectors satisfying the condition per the number of discovered input vectors is 60% for the datasets generated through logistic function.

1 Introduction

As most of the researches in data mining and machine learning has focused on the accuracy, efficiency, and robustness of different techniques, little effort has been made for “actionable knowledge extraction” from advanced machine learning models. Here, “actionable knowledge extraction” means finding how to change input vectors to improve the objective variables in supervised learning; improving the objective variable in regression tasks. However, in the real world, we often

H. Tanaka—Presently with Research Laboratory, NTT DOCOMO Inc.

© Springer Nature Switzerland AG 2018

C. Ordonez and L. Bellatreche (Eds.): DaWaK 2018, LNCS 11031, pp. 216–228, 2018.

https://doi.org/10.1007/978-3-319-98539-8_17

face the difficulty of extracting such knowledge. In other words, we cannot obtain the knowledge to answer the research question “How do we modify the input vectors to increase the objective variables greater than a given threshold or decrease them less than the threshold with minimum effort?”

Let us consider a running example. We are strategists of a soccer team. It is supposed that the winning percentage is predicted by random forest regression whose objective variables are abilities of each position: speed, acceleration, dash speed, and so on. If we can acquire the desirable input variables corresponding to winning percentage which is upper than a given threshold—for example, we want to keep the winning percentage upper than 80%, the threshold is 80; we can plan the training so that the abilities of players will satisfy the desirable input variables. In addition, if the desirable input variables corresponding to the winning percentage which is upper than the threshold is far from the original input variables, we cannot develop the good plan—it is difficult to improve the players’ abilities exponentially. Therefore, the requirement is rewritten as “We want to know the desirable abilities of each player and the abilities are nearest to original abilities of players.”

In this paper, we propose a method named, the Transforming-feAture Method (TRANS-AM) to answer the research question in a regression task. This method enables us to modify the input vector adequately to increase or decrease the objective variables by adding a small positive number ε to each input variable in the original input variables. With the property of a regression tree—it is supposed that the objective variable is predicted by random forest regression [1]—, splitting the input space into subspaces, allows us to determine the input vector whose objective variable improved. By transforming the input vector to new input vectors belonging to one of the subspaces, we discover a new input vector whose distance from the original input vector is minimum in those new vectors. The basic idea of the TRANS-AM is based on the method proposed by Tolomei *et al.* [2]. Their method is built for classification tasks, and we expanded the method for regression tasks. We also relax a restriction of a regression tree which is supposed in the previous study by Tolomei *et al.* [2]. Considering the previous running example, the TRANS-AM enables us to plan the ideal training.

The contributions of this paper are as follows.

- We expanded the classification task discussed in Tolomei *et al.* [2] to the regression task.
- We relaxed a restriction which is assumed in Tolomei *et al.* [2]: once we use the input variable x_i for a branch of tree we cannot use it for other branches.

The second contributions allows us to use more complex regression trees for the random forest more than that used in Tolomei *et al.* [2].

2 Related Work

We first discuss earlier studies on extracting actionable knowledge. Cao *et al.* [3] and Robert *et al.* [4] focused on the development of effective interestingness

metrics. Liu *et al.* [5,6] proposed methods for pruning and summarizing the learnt rules, as well as matching rules by similarity. Cao *et al.* [7,8] proposed a data mining method which is a paradigm shift from a research-centred discipline to a practical tool for actionable knowledge. All the above methods depend on the domains of datasets, but our proposed method does not.

We now discuss tree-based methods. Du *et al.* [9], Karim and Rahman [10], and Yang *et al.* [11,12] discussed post-proceeding methods specifically tailored to decision trees. It is true that a decision tree is interpretable; therefore, we can find a modification approach to improve the objective variable. However, a decision tree's prediction precision is not good [13]. Our proposed method, on the other hand, can use random forest whose prediction precision is better than that of a decision tree.

Finally, Cui *et al.* [14] proved that the optimal action extraction (OAE) problem similar to the problem we solve in Sect. 3 is generally NP-hard by reducing it to DNF-MAXSAT [15] and formulated the problem in an integer linear programming formulation, which has been efficiently solved using current packages with state-of-the-art solvers such as CPLEX [16]. Tolomei *et al.* [2] developed a method of solving the OAE problem with an ε -satisfactory instance, which is explained in Sect. 3. We call this method as actionable feature tweaking (AFT). As mentioned in Sect. 1, the AFT requires the restriction: once we use the input variable x_i for a branch, we cannot use it for other branches. By this restriction, the modification of input vectors is simplified. As we relax the restriction, we change the modification approach, i.e. the approach to developing an ε -satisfactory instance.

3 TRANS-AM: Proposed Method

In this section, we explain the TRANS-AM for feature transformation to increase objective variables more than a given threshold or decrease them than the threshold by expanding AFT. Actionable feature tweaking is used to change the label of an objective variable, i.e. the task addressed in AFT is *classification*. We expanded AFT to change the objective variable to *regression* and liberalize one assumption of AFT regarding the root-to-leaf paths of each regression tree. We published the source of TRANS-AM on <https://github.com/setten-QB/TRANS-AM.git>

3.1 Notation

Let $\mathcal{X} \subset \mathbb{R}^d$ be an input space and suppose that each $\mathbf{x} \in \mathcal{X}$ is associated with an objective variable $y \in \mathcal{Y} \subset \mathbb{R}$. We assume there is an unknown target function $f : \mathcal{X} \rightarrow \mathcal{Y}$. Most machine learning methods learn the function $f \approx \hat{f}$ from dataset $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$. Specifically, \hat{f} is the estimate that best approximates f on \mathcal{D} , according to a specific loss function ℓ . The ℓ measures the error between predicted and observed values.

The interpretability of \hat{f} depends on the hypothesis space in which \hat{f} was selected. In this study, we focused on random forest regression. Random forest regressor \mathcal{T} consists of K regression trees T_1, \dots, T_K . We represent the estimate of each T_k , $k = 1, \dots, K$ as \hat{h}_k . Then the estimate of \mathcal{T} is calculated by the sample mean of \hat{h}_k .

Regression tree T_k splits the input space \mathcal{X} into subspaces as

$$\mathcal{X} = \mathcal{X}_{k,1} \oplus \mathcal{X}_{k,2} \oplus \dots \oplus \mathcal{X}_{k,M}, \quad (1)$$

and $\gamma_{k,m}$ corresponds to the area $\mathcal{X}_{k,m}$. Then the prediction with T_k is calculated by

$$\hat{h}_k(\mathbf{x}) = \sum_{m=1}^M \gamma_{k,m} \mathbf{1}[\mathbf{x} \in \mathcal{X}_{k,m}], \quad \mathbf{1}[\mathbf{x} \in \mathcal{X}_{k,m}] = \begin{cases} 1 & \mathbf{x} \in \mathcal{X}_{k,m} \\ 0 & \mathbf{x} \notin \mathcal{X}_{k,m} \end{cases}. \quad (2)$$

where $\mathbf{1}[\mathbf{x} \in \mathcal{X}_{k,m}]$ means the indicator function which is defined as

$$\mathbf{1}[\mathbf{x} \in \mathcal{X}_{k,m}] = \begin{cases} 1 & \mathbf{x} \in \mathcal{X}_{k,m} \\ 0 & \mathbf{x} \notin \mathcal{X}_{k,m} \end{cases}. \quad (3)$$

3.2 Split Input Space with Regression Tree

Suppose that we are given the trained random forest regressor \mathcal{T} and \mathbf{x} satisfying $f(\mathbf{x}) = \hat{f}(\mathbf{x}) < t_0$, where t_0 is a hyperparameter meaning a lower threshold. Our aim is to transform \mathbf{x} to \mathbf{x}' , which satisfies $f(\mathbf{x}') \geq t_1$, where t_1 is also a hyperparameter meaning an upper threshold.

We assume that Assumption 1 holds for any fixed regression tree T_k .

Assumption 1. For any fixed regression tree T_k of the random forest \mathcal{T} , 4 holds.

$$\exists \mathcal{X}_{k,m} \subset \mathcal{X} \quad \text{s.t.} \quad \forall \mathbf{x} \in \mathcal{X}_{k,m}, \hat{h}_k(\mathbf{x}) = \gamma_{k,m} \geq t_1 \quad (4)$$

By this assumption, we can select the subspace $\mathcal{X}_{k,m}^{t_1}$ whose $\gamma_{k,m}$ satisfies $\gamma_{k,m} \geq t_1$, and $\mathcal{X}_{k,m}^{t_1}$ is written as

$$\mathcal{X}_{k,m}^{t_1} = \prod_{i=1}^d [\theta_i^{\text{low}}, \theta_i^{\text{upp}}], \quad \theta_i^{\text{low}}, \theta_i^{\text{upp}} \in \mathbb{R} \cup \{-\infty, \infty\}. \quad (5)$$

In (5), θ_i^{low} and θ_i^{upp} are decided by the random forest algorithm. In AFT, we have to assume Assumption 2.

Assumption 2 (In AFT). The path $p_{k,m}$ of regression tree T_k from root to m -th leaf is represented as

$$p_{k,m} = \{(x_1 \underset{\geq}{\theta}_1), (x_2 \underset{\leq}{\theta}_2), \dots, (x_d \underset{\geq}{\theta}_d)\}. \quad (6)$$

Assumption (6) means that for all i either $\theta_i^{\text{low}} = -\infty$ or $\theta_i^{\text{upp}} = \infty$ holds. However, with the TRANS-AM, we do not adopt Assumption 2 because in many actual cases the learned regression tree does not satisfy this assumption. Of course we can build a regression tree by satisfying Assumption 2, but it is a little messy and sacrifices prediction flexibility.

3.3 ε -Satisfactory Instance

Let \mathcal{X}_k be the family of all $\mathcal{X}_{k,m}^{t_1}$ in T_k ;

$$\mathcal{X}_k = \{ \mathcal{X}_{k,m} \mid \forall \mathbf{x} \in \mathcal{X}_{k,m}, \hat{h}_k(\mathbf{x}) = \gamma_{k,m} \geq t_1 \}, \quad (7)$$

where $\mathcal{X}_{k,m}$ satisfying $\forall \mathbf{x} \in \mathcal{X}_{k,m}, \hat{h}_k(\mathbf{x}) = \gamma_{k,m} \geq t_1$ is $\mathcal{X}_{k,m}^{t_1}$. Then, $|\mathcal{X}_k|$ is often greater than 1. For all subspaces $\mathcal{X}_{k,m}^{t_1} \in \mathcal{X}$, we build the ε -satisfactory instances $\mathbf{x}_{k(\varepsilon)}^{t_1}$ as following.

$$\mathbf{x}_{k(\varepsilon)}^{t_1}[i] = \begin{cases} \theta_i^{\text{uPP}} - \varepsilon & \theta_i^{\text{low}} = -\infty \\ \theta_i^{\text{low}} + \varepsilon & \theta_i^{\text{uPP}} = \infty \\ \frac{\theta_i^{\text{low}} + \theta_i^{\text{uPP}}}{2} & \text{otherwise} \end{cases} \quad (8)$$

In (8), ε means the distance between $\mathbf{x}_{k(\varepsilon)}^{t_1}$ and the boundaries of subspace, i.e., the position of $\mathbf{x}_{k(\varepsilon)}^{t_1}$ is determined by ε if $\theta_i^{\text{low}} = -\infty$ or $\theta_i^{\text{uPP}} = \infty$.

The ε -satisfactory instance defined by (8) belongs to $\mathcal{X}_{k,m}^{t_1} \in \mathcal{X}$. If $\mathbf{x}_{k(\varepsilon)}^{t_1}[i] = \theta_i^{\text{uPP}} - \varepsilon$, the i -th element of $\mathbf{x}_{k(\varepsilon)}^{t_1}$ is the point transformed by $-\varepsilon$ from θ_i for the negative direction parallel to the x_i axis (or vice versa). If $\mathbf{x}_{k(\varepsilon)}^{t_1}[i] = (\theta_i^{\text{low}} + \theta_i^{\text{uPP}}) / 2$, the i -th element of $\mathbf{x}_{k(\varepsilon)}^{t_1}$ is the centre of interval $[\theta_i^{\text{low}}, \theta_i^{\text{uPP}}]$.

In any case $\mathbf{x}_{k(\varepsilon)}^{t_1} \in \mathcal{X}_{k,m}^{t_1}$ holds; therefore, $\hat{h}(\mathbf{x}_{k(\varepsilon)}^{t_1}) \geq t_1$ also holds.

3.4 Feature Transformation

Let $\mathcal{X}_{\cdot}^{t_1}$ be the set of all ε -satisfactory instances. More specifically, $\mathcal{X}_{\cdot}^{t_1}$ is written by

$$\mathcal{X}_{\cdot}^{t_1} = \bigcup_{k=1}^K \bigcup_{m: \mathcal{X}_{k,m}^{t_1} \in \mathcal{X}} \mathbf{x}_{k(\varepsilon)}^{t_1}, \quad (9)$$

where $\mathbf{x}_{k(\varepsilon)}^{t_1}$ depends on subspace $\mathcal{X}_{k,m}^{t_1}$. Therefore, $\bigcup_{m: \mathcal{X}_{k,m}^{t_1} \in \mathcal{X}} \mathbf{x}_{k(\varepsilon)}^{t_1}$ means the set of all the ε -satisfactory instances made using T_k . Then the transformed input vector we want is given by solving the following optimization problem.

$$\mathbf{x}' = \arg \min_{\mathbf{x}_{k(\varepsilon)}^{t_1} \in \mathcal{X}_{\cdot}^{t_1} \mid \hat{f}(\mathbf{x}_{k(\varepsilon)}^{t_1}) \geq t_1} \delta(\mathbf{x}, \mathbf{x}_{k(\varepsilon)}^{t_1}) \quad (10)$$

In (10), cost function δ means the distance between \mathbf{x} and $\mathbf{x}_{k(\varepsilon)}^{t_1}$. With this δ , we choose optimal vector \mathbf{x}' , i.e. the selected \mathbf{x}' is nearest to the original input vector. As we explained in Sect. 1, Cui *et al.* [14] proved that the OAE problem, which is essentially identical to (10), is generally NP-hard, and the TRANS-AM translates the OAE problem into a closed-form integer linear programming (ILP) problem, which can be efficiently solved by off-the-shelf ILP solvers. On

the other hand, Tolomei *et al.* [2] introduced an algorithm to obtain \mathbf{x}' by using an ε -satisfactory instance.

In this paper, we used an expanded algorithm of Tolomei *et al.* [2] to discovery \mathbf{x}' . We show the algorithm for finding \mathbf{x}' in Algorithm 1. This algorithm seeks \mathbf{x}' which is solution of the following optimization problem:

$$\mathbf{x}' = \underset{\mathbf{x}_{k(\varepsilon)}^{t_1} \in \mathcal{X}^{t_1}}{\operatorname{arg\,min}} \delta(\mathbf{x}, \mathbf{x}_{k(\varepsilon)}^{t_1}) \quad (11)$$

In (11), we dropped the condition $\hat{f}(\mathbf{x}_{k(\varepsilon)}^{t_1}) \geq t_1$ of (10), because the condition is implicitly satisfied by definition of ε -satisfactory instance. Algorithm 1 often returns $\mathbf{x}' = \mathbf{x}$ because sometimes all the ε -satisfactory instances $\mathbf{x}_{j(\varepsilon)}^{t_1}$ built with Algorithm 1 do not satisfy the 8-th line if-statement $\hat{f}(\mathbf{x}_{j(\varepsilon)}^{t_1}) \geq t_1$. Hence we should evaluate the TRANS-AM carefully in Sect. 4.

Algorithm 1. Algorithm of TRANS-AM

Require: $\mathcal{T} = \{T_1, T_2, \dots, T_K\}$, thresholds t_0, t_1 , input vector \mathbf{x} such that $f(\mathbf{x}) < t_0$, cost function δ , and $\varepsilon > 0$

Ensure: \mathbf{x}' satisfying $\hat{f}(\mathbf{x}') \geq t_1$

```

1:  $\mathbf{x}' \leftarrow \mathbf{x}$ 
2:  $\delta_{\min} \leftarrow \infty$ 
3: for  $k = 1, 2, \dots, K$  do
4:   if  $\hat{f}(\mathbf{x}) < t_1 \wedge \hat{h}_k(\mathbf{x}) < t_1$  then
5:     make  $\mathcal{X}_k$ 
6:     for  $\mathcal{X}_{k,m}^{t_1} \in \mathcal{X}_k$  do
7:       build  $\varepsilon$ -satisfactory instance  $\mathbf{x}_{j(\varepsilon)}^{t_1}$ 
8:       if  $\hat{f}(\mathbf{x}_{j(\varepsilon)}^{t_1}) \geq t_1$  then
9:         if  $\delta(\mathbf{x}, \mathbf{x}_{j(\varepsilon)}^{t_1}) < \delta_{\min}$  then
10:            $\mathbf{x}' \leftarrow \mathbf{x}_{j(\varepsilon)}^{t_1}$ 
11:            $\delta_{\min} \leftarrow \delta(\mathbf{x}, \mathbf{x}_{j(\varepsilon)}^{t_1})$ 
12:         end if
13:       end if
14:     end for
15:   end if
16: end for
17: return  $\mathbf{x}'$ 

```

4 Numerical Simulation and Evaluation

In this section, we explain the results of numerical simulations. The aim with TRANS-AM is to transform the input vector \mathbf{x} with $\hat{f}(\mathbf{x}) < t_0$ to \mathbf{x}' satisfying

$f(\mathbf{x}') \geq t_1$. If the random forest estimates the unknown target function f , Algorithm 1 can return \mathbf{x}' such that $f(\mathbf{x}') \geq t_1$. However, in practice the random forest cannot estimate f completely, so the vector \mathbf{x}' yielded from Algorithm 1 does not always satisfy $f(\mathbf{x}') \geq t_1$, i.e. Algorithm 1 sometimes yields \mathbf{x}' satisfying not $f(\mathbf{x}') \geq t_1$ but $\hat{f}(\mathbf{x}') \geq t_1$. Therefore, we should evaluate how many vectors yielded from Algorithm 1 satisfy $f(\mathbf{x}') \geq t_1$. For such an evaluation, we should know the target function f ; hence, we evaluated the TRANS-AM not through application for real datasets but numerical simulations.

As we mentioned in Sect. 3, Algorithm 1 often returns $\mathbf{x}' = \mathbf{x}$. Therefore, we use (18) as an indicator for evaluating this problem.

4.1 Experimental Setting

We evaluated the TRANS-AM with artificial data. The artificial datasets were generated in the following steps, where $\mathbf{1}_d$ is the d -dimensional vector whose components are 1, I_d is the $d \times d$ diagonal matrix whose diagonal components are 1, and $\mathcal{N}_d(\mathbf{1}_d, I_d)$ means the d -dimensional Gaussian distribution whose mean vector is $\mathbf{1}_d$ and covariance matrix is I_d .

Step 1. generate $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}_d(\mathbf{1}_d, I_d)$.

Step 2. make independent variable y_1, y_2, \dots, y_N by $y_n = f(\mathbf{x}_n) + \eta_n$, $\eta_n \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, 1)$

Note that f in the above steps is the same as the unknown target function in Sect. 3. In our simulation, we used the following functions as $f(\mathbf{x})$.

$$f_1(\mathbf{x}) = \mathbf{a}^T \mathbf{x} \quad (12)$$

$$f_2(\mathbf{x}) = \sin(\mathbf{a}^T \mathbf{x}) \quad (13)$$

$$f_3(\mathbf{x}) = \sum_{i=1}^n a_i \sin x_i \quad (14)$$

$$f_4(\mathbf{x}) = \exp(\mathbf{a}^T \mathbf{x}) \quad (15)$$

$$f_5(\mathbf{x}) = \frac{1}{1 + \exp(\mathbf{a}^T \mathbf{x})} \quad (16)$$

In functions (12)–(16), $\mathbf{a} \sim \mathcal{N}(\mathbf{0}, I)$ and a_i is the i -th element of \mathbf{a} . The TRANS-AM has the parameters t_0 , t_1 and ε . Parameters t_0 and t_1 are determined by analysts according to their aim, and ε should be turned using some kind of data-driven method. The following steps comprise the simulation process.

step 1. split the dataset \mathcal{D} into training set $\mathcal{D}_{\text{train}} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N_0}$ and test set

$$\mathcal{D}_{\text{test}} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N_1}, \text{ where } N = N_0 + N_1$$

step 2. let t_1 be the p_{upp} percentile point of $y \in \mathcal{D}_{\text{train}}$ and let t_0 be the p_{low} percentile point of $y \in \mathcal{D}_{\text{train}}$.

step 3. fix ε to one of candidates of ε and train the random forest regressor using $\mathcal{D}_{\text{train}}$.

- step 4. choose the input vectors in $\mathcal{D}_{\text{test}}$ whose objective variables are less than t_0 .
- step 5. transform the input vectors to the new input vectors \mathbf{x}' with the TRANS-AM.
- step 6. evaluate the TRANS-AM by using three criterion that are shown in (17), (18), and (19).

Score P represents how many input vectors are transformed using the TRANS-AM regardless of whether \mathbf{x}' satisfies $f(\mathbf{x}') \geq t_1$, score Q indicates how many input vectors are modified per number of input vectors, and score R represents how many modified input vectors $\mathbf{x}'(\neq \mathbf{x})$ satisfy $f(\mathbf{x}') \geq t_1$ per number of changed input vectors. Algorithm 1 often yields the same vector as an input vector. Therefore, we evaluated how many yielded vectors satisfy our purpose $f(\mathbf{x}') \geq t_1$ per number of transformed vectors with the score R .

$$P = \frac{|\{\mathbf{x}' \mid f(\mathbf{x}') \geq t_1\}|}{|\{\mathbf{x} \mid \hat{f}(\mathbf{x}) < t_0\}|} \tag{17}$$

$$Q = \frac{|\{\mathbf{x}' \mid \mathbf{x}' \neq \mathbf{x}\}|}{|\{\mathbf{x} \mid \hat{f}(\mathbf{x}) < t_0\}|} \tag{18}$$

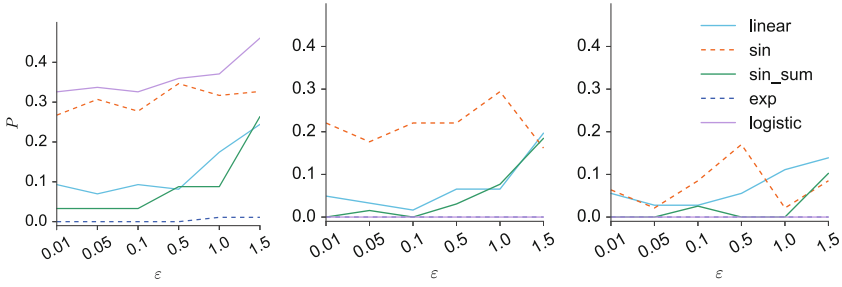
$$R = \frac{|\{\mathbf{x}' \mid f(\mathbf{x}') \geq t_1\}|}{|\{\mathbf{x}' \mid \mathbf{x}' \neq \mathbf{x}\}|} \tag{19}$$

Among these three scores, a relationship $P = QR$ holds. Score P is most noticeable score and scores Q and R construct P . We simulated all combinations of $\varepsilon \in \{0.01, 0.05, 0.1, 0.5, 1, 1.5\}$, $N_0 = 1000$, $N_1 = 250$ and $d = 50$.

4.2 Result and Consideration

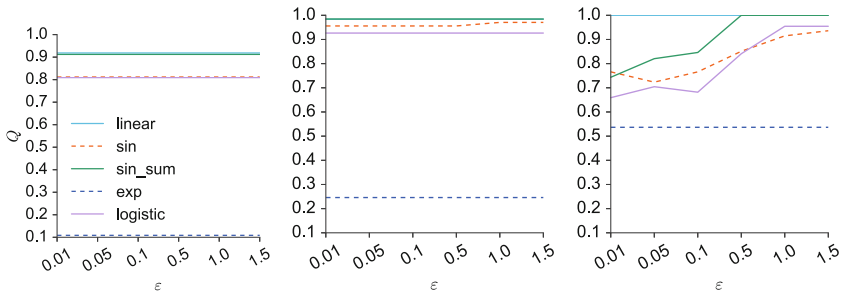
We show the simulation results in Figs. 1, 2 and 3. Figure 1 shows the relationship between ε and score P , Fig. 2 shows the relationship between ε and score Q , and Fig. 3 shows the relationship between ε and score R . In the each figure, *linear*, *sin*, *sinsum*, *exp*, and *logistic* mean the datasets generated by (12), (13), (14), (15), and (16), respectively.

As shown in Fig. 1, for $(p_{\text{low}}, p_{\text{upp}}) = (40, 60)$, the dataset generated by (16) shows the best score P . In addition, (13) shows the second-best score P . This is because (16) and (13) are upper bounded, i.e. $\exists K : \text{const. s.t. } \forall \mathbf{x} \in \mathbb{R}, f(\mathbf{x}) < K$. As we explained in Sect. 3, the regression tree splits the input space into subspaces $\{\mathcal{X}_m\}_{m=1}^M$ then corresponds γ_m with \mathcal{X}_m . The precisions of approximating the functions that are not upper bounded with random forest become worse due to this approximation. For example, suppose that we approximate the function $g(x) = \exp(x)$ and can use the regression tree. Then the input space is divided into M subspaces $\mathcal{X}_1, \dots, \mathcal{X}_M$. The prediction value γ_m of $x \in \mathcal{X}_m$ is generally $(1/|\{x \in \mathcal{X}_m\}|) \sum_{x \in \mathcal{X}_m} x$, where x is the training sample. Therefore, the difference $\|f(x) - \gamma_m\|$ increases as x becomes larger.



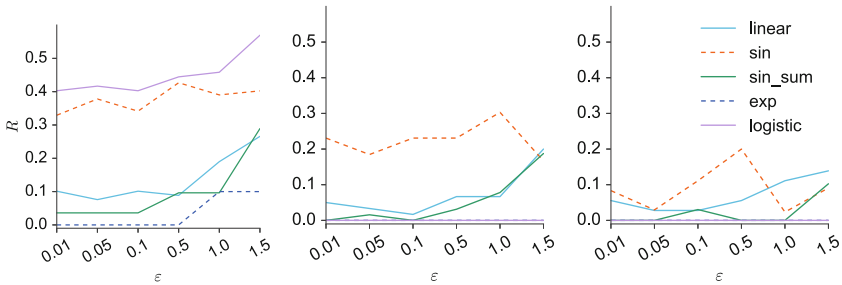
(a) $(p_{low}, p_{upp}) = (40, 60)$ (b) $(p_{low}, p_{upp}) = (30, 70)$ (c) $(p_{low}, p_{upp}) = (20, 80)$

Fig. 1. Relationships between ε and P



(a) $(p_{low}, p_{upp}) = (40, 60)$ (b) $(p_{low}, p_{upp}) = (30, 70)$ (c) $(p_{low}, p_{upp}) = (20, 80)$

Fig. 2. Relationships between ε and Q



(a) $(p_{low}, p_{upp}) = (40, 60)$ (b) $(p_{low}, p_{upp}) = (30, 70)$ (c) $(p_{low}, p_{upp}) = (20, 80)$

Fig. 3. Relationships between ε and R

On the other hand, in Fig. 1(b) and (c), the score P for the dataset generated by (16) becomes worse than that in the case of Fig. 1(a). The distribution of $f(\mathbf{x}')$ is illustrated in Fig. 4. As shown in Fig. 4, the threshold is pulled in the positive direction as percentile point becomes larger. Essentially, the objective variable does not appear in the region upper than 1 due to the range of (16);

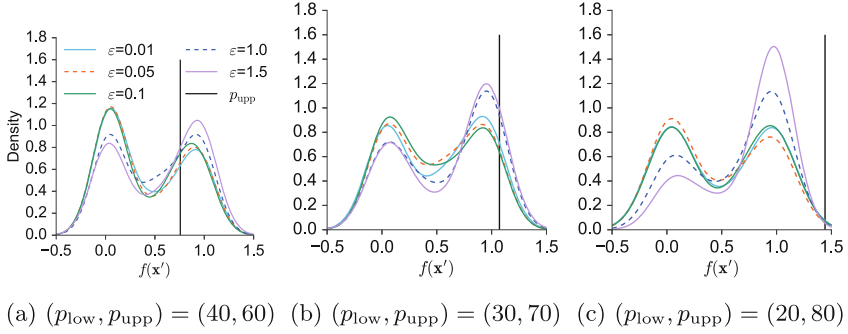


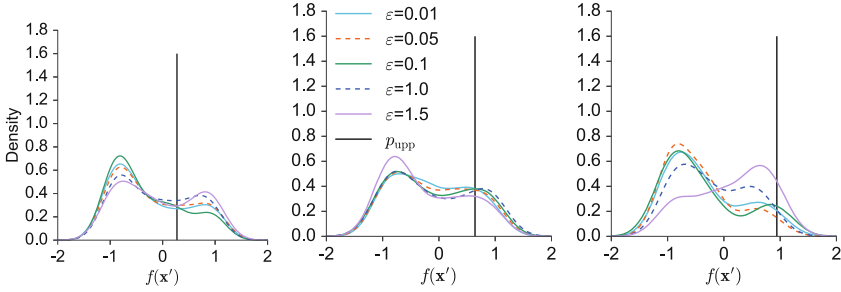
Fig. 4. Distributions of the objective variables generated by (16) and the threshold

we call the range of $f(x)$ “essential range of the objective variable”—the noise η affects the range of the objective variable. The reason why the threshold in the centre and right of Fig. 4 is greater than 1 is affection of noise η . In the case of Fig. 4(b) and (c), the setting of simulation—we aim to transform \mathbf{x} into \mathbf{x}' satisfying $f(\mathbf{x}') \geq p_{upp}$ —is essentially impossible, because the objective variable does not become greater than p_{upp} . Therefore, the results of Fig. 4(b) and (c) are not strange. Videlicet, the settings of the centre and right of Fig. 5 are not suitable, i.e. it is essentially impossible to obtain the transformed input vector \mathbf{x}' satisfying $f(\mathbf{x}') \geq t_{upp}$ because the threshold is out of the essential range of the objective variable.

As shown in Fig. 2(a) and (b), each score R is higher than 0.8, except for *exp*. These results mean that for the datasets generated by (15), Algorithm 1 cannot transform \mathbf{x} into \mathbf{x}' . This is also because random forest cannot approximate Exponential Function (15) well. Of course, other functions (12) and (14) are not upper bounded. However, Exponential Function (15) diverges to infinity earlier than (12) and (14). For $f(x) = \exp(ax)$, the difference between $f(x)$ and $f(x + c)$, where c is a very small positive integer, is larger than that for $f(x) = ax$. Therefore, the approximation precision for the dataset generated by (15) becomes worse than that for the other. If the prediction precision is very bad, the condition $\hat{f}(\mathbf{x}_{j(\epsilon)}^{t_1}) \geq t_1$ in Algorithm 1 cannot be satisfied, i.e. \mathbf{x} is yielded using Algorithm 1. Therefore, we can conclude that for the dataset whose objective variable is generated by the exponential function of the input variable of the TRANS-AM cannot find the modified \mathbf{x}' . In the right figure of Fig. 2, turning the ϵ well, the score Q become higher than 0.8.

As shown in the Fig. 3(a), the score R s for the datasets generated by (13) and (16) can be greater than 0.4 by turning ϵ for $p_{low} = 40, p_{upp} = 60$. In Fig. 3(b) and 1(c), however, the score R for the datasets generated by (16) one of the lowest. The reason is referred to in the discussion of Fig. 1. As p_{low} becomes smaller and p_{upp} becomes larger, the score R decrease.

Figure 5 illustrates the reason that the TRANS-AM performs well for the dataset generated by (13). The thresholds p_{upp} for *sin* are not pulled in the



(a) $(p_{low}, p_{upp}) = (40, 60)$ (b) $(p_{low}, p_{upp}) = (30, 70)$ (c) $(p_{low}, p_{upp}) = (20, 80)$

Fig. 5. Distributions of the objective variables generated by (13) and the threshold

positive direction so strongly, as for *logistic*, i.e. the threshold appears in the essential range of the objective variable. Therefore, the TRANS-AM performed better for the dataset generated by (13) than for the other datasets.

In all of that, the TRANS-AM does not work if the following Case 1 or 2 holds:

- Case 1: the random forest cannot approximate the target function, e.g. the objective variables are generated from an upper unbound function;
- Case 2: the threshold is out of the essential range of the objective variable.

The case 1 corresponds to the case of *exp* (see Figs. 1, 2 and 3) and the case 2 corresponds to the case of *sin* and *logistic* (see the centre and right of Fig. 1 and 3). However, if neither of the two cases holds, TRANS-AM will work well—we can obtain the input vectors satisfying that the objective variables are greater than the threshold from the candidates of input vectors by the score $P \geq 45$.

5 Conclusion

We proposed the TRANS-AM for discovering a new input vector to increase the objective variable greater than a given threshold or decrease it than the threshold with minimizing δ in a regression task. We relaxed the restrictions assumed by Tolomei *et al.* [2] because the restrictions were not reasonable for random forest. As we discussed in the introduction, we are often faced with the question “How do we modify the input vector to increase the objective variable greater than the given threshold or decrease it than the threshold with minimum effort?” The TRANS-AM is an answer to this question and we have disclosed the situation that the method works well.

In the TRANS-AM, we generated candidates of the new input vector by ε -satisfactory instances. The ε -satisfactory instances satisfy the condition, i.e. the objective variables corresponding to the ε -satisfactory instances are greater than the threshold. Then we select the new input vector which minimizes the

distance between the original input vector and new one—we select the new input vector by (11).

Considering the use case of the TRANS-AM, we have to pay attention to what kind of input variables are contained in the input vector. The transformation by TRANS-AM affects all of the input variables generally, because of (8). Therefore, in the use case, we have to use only input variables which can be changed. In the case of example referred in Sect. 1—we are strategists of a soccer team, we can use the *energy*, *speed* of players. However, we cannot use the body size of the players, because we cannot control their values.

We evaluated the TRANS-AM by moving ε and the thresholds— p_{low} and p_{upp} —in Sect. 4. Then we found that for the dataset whose objective variables are generated by the sin formula, the TRANS-AM shows good precision; score P and score R are maximum or pre-maximum in each case of Figs. 1 and 3. On the other hand, for the dataset whose objective variables are generated by (16), the TRANS-AM does not work well except when $(p_{\text{low}}, p_{\text{upp}}) = (40, 60)$. In such a situation, it is hard for TRANS-AM to find the transformed input vector satisfying the condition. These considerations are summarized as the Case 1 and Case 2. If we avoid the two cases, i.e. we can approximate the target function by the random forest and the threshold is not out of the essential range of the objective variable; we obtain the transformed input vector efficiently by using TRANS-AM.

For future works, we plan to extend the TRANS-AM for gradient boosting [17] and XGBoost [18]—these additive tree models are effective predictors for various fields [13, 19–21]. TRANS-AM works well for datasets whose objective variables are generated by the sin formula but does not work well for datasets whose objective variables are generated by an upper unbounded formula (e.g. exp formula). Therefore, a method that works well for upper unbounded datasets is required for discovering new input vectors whose objective variables are larger than a given threshold.

Acknowledgement. This research was partially supported by NAIST Big Data Project.

References

1. Breiman, L.: Random forests. *Mach Learn.* **45**(1), 5–32 (2001)
2. Tolomei, G., Silvestri, F., Haines, A., Lalmas, M.: Interpretable predictions of tree-based ensembles via actionable feature tweaking. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 465–474. ACM (2017)
3. Cao, L., Luo, D., Zhang, C.: Knowledge actionability: satisfying technical and business interestingness. *IJBIDM* **2**, 496–514 (2007)
4. Hilderman, R.J., Hamilton, H.J.: Applying objective interestingness measures in data mining systems. In: Zighed, D.A., Komorowski, J., Żytkow, J. (eds.) *PKDD 2000*. LNCS (LNAI), vol. 1910, pp. 432–439. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45372-5_47

5. Liu, B., Hsu, W.: Post-analysis of learned rules. In: AAAI/IAAI, vol. 1, pp. 828–834 (1996)
6. Liu, B., Hsu, W., Ma, Y.: Pruning and summarizing the discovered associations. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 125–134. ACM (1999)
7. Cao, L., Zhang, C.: Domain-driven, actionable knowledge discovery. *IEEE Intell. Syst.* **22**(4) (2007)
8. Cao, L., Zhao, Y., Zhang, H., Luo, D., Zhang, C., Park, E.K.: Flexible frameworks for actionable knowledge discovery. *IEEE Trans. Knowl. Data Eng.* **22**(9), 1299–1312 (2010)
9. Du, J., Hu, Y., Ling, C.X., Fan, M., Liu, M.: Efficient action extraction with many-to-many relationship between actions and features. In: van Ditmarsch, H., Lang, J., Ju, S. (eds.) LORI 2011. LNCS (LNAI), vol. 6953, pp. 384–385. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24130-7_29
10. Karim, M., Rahman, R.M.: Decision tree and naive bayes algorithm for classification and generation of actionable knowledge for direct marketing. *J. Softw. Eng. Appl.* **6**(04), 196 (2013)
11. Yang, Q., Yin, J., Ling, C., Pan, R.: Extracting actionable knowledge from decision trees. *IEEE Trans. Knowl. Data Eng.* **19**(1), 43–56 (2007)
12. Yang, Q., Yin, J., Ling, C.X., Chen, T.: Postprocessing decision trees to extract actionable knowledge. In: Third IEEE International Conference on Data Mining, ICDM 2003, pp. 685–688. IEEE (2003)
13. Friedman, J., Hastie, T., Tibshirani, R.: The elements of statistical learning. Springer Series in Statistics, vol. 1. Springer, New York (2001). <https://doi.org/10.1007/978-0-387-21606-5>
14. Cui, Z., Chen, W., He, Y., Chen, Y.: Optimal action extraction for random forests and boosted trees. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 179–188. ACM (2015)
15. Manindra, A., Thomas, T.: Satisfiability problems. Technical report (2000)
16. CPLEX, I.I.: V12. 1: User’s manual for cplex. *Int. Bus. Mach. Corp.* **46**(53), 157 (2009)
17. Friedman, J.H.: Stochastic gradient boosting. *Comput. Stat. Data Anal.* **38**(4), 367–378 (2002)
18. Chen, T., Guestrin, C.: Xgboost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery And Data Mining, pp. 785–794. ACM (2016)
19. Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., Cook, M., Moore, R.: Real-time human pose recognition in parts from single depth images. *Commun. ACM* **56**(1), 116–124 (2013)
20. Viola, P., Jones, M.J.: Robust real-time face detection. *Int. J. Comput. Vis.* **57**(2), 137–154 (2004)
21. Tyree, S., Weinberger, K.Q., Agrawal, K., Paykin, J.: Parallel boosted regression trees for web search ranking. In: Proceedings of the 20th International Conference on World Wide Web, pp. 387–396. ACM (2011)

Sequences



Discovering Periodic Patterns Common to Multiple Sequences

Philippe Fournier-Viger¹(✉), Zhitian Li², Jerry Chun-Wei Lin³,
Rage Uday Kiran⁴, and Hamido Fujita⁵

¹ School of Natural Sciences and Humanities,
Harbin Institute of Technology, Shenzhen, China
philfv8@yahoo.com

² School of Computer Sciences and Technology,
Harbin Institute of Technology, Shenzhen, China
tymonlee1@163.com

³ Department of Computing, Mathematics and Physics,
Western Norway University of Applied Sciences (HVL), Bergen, Norway
jerrylin@ieee.org

⁴ Institute of Industrial Science University of Tokyo, Tokyo, Japan
uday.rage@gmail.com

⁵ Faculty of Software and Information Science,
Iwate Prefectural University, Takizawa, Iwate, Japan
hfujita-799@acm.org

Abstract. Discovering periodic itemsets in transaction databases is an emerging data mining task. However, current algorithms are designed to discover periodic itemsets in a single sequence. But in real-life, it is desirable to find periodic patterns that are common to multiple sequences. For example, a retail store manager can benefit from finding that many customers buy the same products every week in a retail store, to adapt its marketing and sale strategies. To address this drawback of previous work, this paper defines the problem of mining periodic patterns common to multiple sequences and proposes an efficient algorithm named MPFPS, which relies on a novel PFPS-list structure and two novel periodicity measures to assess periodicity with more flexibility. Experiments on several synthetic and real-life databases show that MPFPS is efficient and can filter many non-periodic itemsets to reveal the desired patterns.

Keywords: Frequent pattern · Periodic pattern · Sequence database

1 Introduction

Frequent pattern mining consists of discovering patterns that appear frequently in a database. It plays an important role in data mining and has numerous applications [3, 10]. Several work in frequent pattern mining have focused on discovering patterns such as frequent itemsets and association rules that do not

consider the sequential ordering of data. But for several applications, considering the sequential ordering is meaningful and can reveal interesting information. For example, when analyzing customer behavior, it can be discovered that some actions are repeated by several customers over time. Discovering such information can be used to design effective marketing and sales strategies.

To discover patterns in data while considering the sequential ordering, itemset mining has been generalized as the problem of sequential pattern mining. It consists of discovering frequent subsequences in a set of sequences [1, 14]. Although sequential pattern mining has numerous applications and several algorithms have been designed, an important limitation is that they are inappropriate to discover recurring patterns. But recurring patterns are found in many domains [6, 15]. For example, one may detect recurring customer behavior such that some customers buy some products every day, week, or month.

To discover recurring patterns, also called periodic patterns, several algorithms have been proposed in recent years [6, 8, 9, 11–13]. However, most algorithms for discovering periodic patterns are designed to find patterns in a single sequence. But periodic patterns also commonly appear in sequence databases (a set of multiple sequences). For example, it is desirable to discover periodic behavior of not just one but common to several customers. To our best knowledge, only one algorithm, named PHUSPM, was proposed to mine periodic patterns in a sequence database [16]. However, this algorithm simply applies the same periodicity measures as algorithms for discovering patterns in a single sequence. As a result, PHUSPM can find patterns that regularly appear in a sequence database, but it does not consider whether these patterns are periodic in each sequence. But finding patterns that are periodic in many sequences is useful. For example, consider sequences of customer transactions in a retail store. The PHUSPM algorithm could find that bread and milk are periodically sold by the store (appear periodically in the database) but would fail to find that many customers periodically buy bread and milk (bread and milk are periodic in multiple sequences, each corresponding to a customer). Finding such information is useful for designing effective sale and marketing strategies.

To address the above limitations of previous work, this paper proposes the task of mining Periodic Frequent Patterns common to multiple Sequences (PFPS), which considers the periodicity of patterns in each sequence and their frequency in the overall database. Moreover, an efficient algorithm is presented to mine PFPS. The contributions of this paper are as follows:

- The problem of mining periodic frequent patterns common to multiple sequences is defined, and its properties are studied.
- To evaluate the periodicity of patterns in each sequence, a new measure is defined, which is the standard deviation of periods, to find patterns that occur with regularity. Moreover, a novel periodicity measure named *Sequence Periodic Ratio* (SPR) is defined to find patterns that are periodic in multiple sequences. To effectively reduce the search space, an upper-bound on the SPR called *boundRa* is developed and a novel pruning property is proposed.

- An algorithm named MPFPS is presented to efficiently find all periodic frequent patterns common to multiple sequences, which relies on a novel PFPS-list structure to avoid repeatedly scanning the database.
- An experimental evaluation on several real and synthetic datasets reveal that the proposed algorithm is efficient and can filter many non periodic patterns.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 defines the problem of periodic frequent pattern mining. Section 4 presents the proposed algorithm. Section 5 describes the experimental evaluation. Finally, Sect. 6 draws the conclusion and discusses future work.

2 Related Work

Several Frequent Itemset Mining (FIM) algorithms have been proposed to discover frequent itemsets, i.e. sets of items that appear frequently in a customer transaction database [2]. One of the most influential, named Apriori explores the search space of itemsets using a breadth-first search. It combines pairs of itemsets to generate larger candidate itemsets, and then scans the database to calculate the support of itemsets and eliminate infrequent itemsets. However, repeated database scans result in poor performance for large databases. The Eclat [4] algorithm addresses this issue by creating a structure called *id-list* for each considered itemset, which can be built from other id-lists to avoid performing database scans. Eclat performs a depth-first search and divides the search space into equivalence classes [4]. To consider the sequential ordering between transactions, FIM algorithms have been extended for the task of Sequential Pattern Mining (SPM), which consists of discover subsequences appearing in many sequences. Some representative SPM algorithms are AprioriAll [1], and PrefixSpan [5]. The former extends the Apriori algorithm.

Recently, FIM has been extended to discover periodic patterns in a single sequence. A frequent itemset is said to be periodic in a sequence if it appears multiple times and the time between each occurrence is less than a user-defined maximum periodic threshold. More efficient algorithms have been designed, and variation of the problem of mining periodic patterns in a single sequences have been proposed [11]. For example, a study [11] addressed the “rare item problem” by proposing to define a minimum support threshold for each item rather than using the same threshold for all items. In another study, an algorithm named *MTKPP* was proposed to discover the k periodic patterns that are the most frequent in a sequence [13]. However, a drawback of all these studies is that the maximum periodicity constraint is very strict. If a pattern appears regularly in a database but appear a single time with a time interval larger than the maximum periodicity threshold, it is discarded. Thus more flexible measures are needed [9]. Although much work has been done to find periodic patterns in a sequence, to the best of our best knowledge, only one algorithm named PHUSPM [16] considers discovering patterns in multiple sequences. But to do that, it considers multiple sequences as a sequence, and then simply applies the same periodicity measures designed for a single sequence. As a result, the algorithm does not consider

whether a pattern is periodic in each sequence. Thus, PHUSPM is unable to find patterns such that several customers periodically buy some products every week. This paper addresses this issue by proposing a more general model and algorithm for mining periodic patterns that are periodic in multiple sequences.

3 Definitions and Problem Statement

This section is divided into two parts. It first presents the problem of discovering periodic patterns in a single sequence, and proposes a novel measure called the periodic standard deviation to filter non interesting periodic patterns. Then, the problem of periodic pattern mining is generalized to mine patterns in a sequence database (multiple sequences).

Table 1. An example sequence database

Sequence_id	Sequence
1	$\langle (a, b, e), (a, b, e), (a, d), (a, e), (a, b, c) \rangle$
2	$\langle (c), (a, b, c, e), (c, d), (a, b, c, e), (a, b, d) \rangle$
3	$\langle (b, c), (a, b), (a, c, d), (a, c), (a, b) \rangle$
4	$\langle (a, b, d, e), (a, b, e), (a, b, c), (a, b, d, e), (a, b) \rangle$

Let there be a set of items I representing all the symbols in a database. An itemset X is a subset of I , that is $X \subseteq I$. An itemset containing k items is said to be a k -itemset. A sequence s is an ordered list of itemsets $s = \langle T_1, T_2, \dots, T_m \rangle$, where $T_j \subseteq I$ ($1 \leq j \leq m$), j is the transaction identifier of T_j , and T_j is said to be a transaction. A sequence database D is an ordered set of n sequences, denoted as $D = \langle s_1, s_2, \dots, s_n \rangle$. The sequence s_i in D is said to be the i -th sequence of D , and its sequence identifier is said to be i . A sequence $s_a = \langle A_1, A_2, \dots, A_k \rangle$ is said to be a subsequence of a sequence $s_b = \langle B_1, B_2, \dots, B_l \rangle$ iff there exist integers $1 \leq i_1 < i_2 < \dots < i_k \leq m$ such that $A_1 \subseteq B_{i_1}, A_2 \subseteq B_{i_2}, \dots, A_k \subseteq B_{i_k}$ (denoted as $s_a \sqsubseteq s_b$). For example, consider the database of Table 1, containing four sequences, which will be used as running example. The first sequence contains five itemsets. The first itemset contains three items (a , b and c). It is thus a 3-itemset. The sequence $\langle (a, b), (a) \rangle$ is a subsequence of s_1 . In previous work, to find periodic patterns in a single sequence, the concept of periods was introduced, which is defined as follows [9].

Definition 1 (periods of an itemset in a sequence). Consider a sequence s_i of a database D and an itemset X . Let $TR(X, s_i) = \langle T_{g_1}, T_{g_2}, \dots, T_{g_k} \rangle \sqsubseteq s_i$ be the ordered set of transactions in which itemset X occurs in sequence s_i . Two transactions T_x and T_y in s_i are said to be consecutive with respect to X if there does not exist a transaction $T_z \in s_i$ such that $x < z < y$ and $X \subseteq T_z$. The period of two consecutive transactions T_x and T_y for a pattern X is $per(T_x, T_y) = y - x$.

The periods of X in a sequence s_i are $pr(X, s_i) = \{per_1, per_2, \dots, per_{k+1}\}$ where $per_1 = g_1 - g_0, per_2 = g_2 - g_1, \dots, per_{k+1} = g_{k+1} - g_k$, where g_1, g_2, \dots, g_k are the numbers of the transactions itemset X occurs and g_0 and g_{k+1} are defined as $g_0 = 0$ and $g_{k+1} = n$, respectively.

For example, the itemset $\{a, b\}$ occurs in transactions T_1, T_2 and T_5 of sequence s_1 . Thus $TR(\{a, b\}, s_1) = \{T_1, T_2, T_5\}$ and the periods of pattern $\{a, b\}$ are $pr(\{a, b\}, s_1) = \{1, 1, 3, 0\}$.

The most widely used measure to assess the periodicity of a pattern in a single sequence is the maximum periodicity [9]:

Definition 2 (maximum periodicity). The maximum periodicity of an itemset X in a sequence s is defined as $maxPr(X, s) = argmax(pr(X, s))$.

In previous work, a pattern is deemed periodic if its maximum period is smaller than a user-defined $maxPr$ threshold, and its support is no less than a threshold $minSup$. The support of an itemset X in a sequence s is the number of transactions containing X in s , that is $sup(X, s) = |TR(X, s)|$. For example, consider $maxPr = 3$ and $minSup = 3$. The itemset $\{a, b\}$ is periodic in sequence s_1 since its periods in that sequence are $pr(\{a, b\}, s_1) = \{1, 1, 3, 0\}$, its maximum period is $maxPr(\{a, b\}, s_1) = max\{1, 1, 3, 0\} = 3 \leq maxPr$ and $sup(\{a, b\}, s_1) = 3 \geq minSup$. However, a problem of $maxPr$ measure is that if it is set to a small value, patterns may be discarded if they only have a few periods greater than $maxPr$, while if $maxPr$ is set to a large value, patterns having periods that vary greatly may be considered as periodic. To address this problem, this paper proposes to consider the standard deviation of periods.

Definition 3 (standard deviation of periods). The standard deviation of the periods of an itemset X in a sequence s is denoted as $stanDev(X, s)$.

For example, the itemset $\{a, b\}$ has four periods in sequence s_1 , that is $pr(X) = \{1, 1, 3, 0\}$. The average period of $\{a, b\}$ is: $avgPr(\{a, b\}, s_1) = (1 + 1 + 3 + 0)/4 = 1.25$. The standard deviation is calculated as: $stanDev(\{a, b\}, s_1) = \sqrt{[(1 - 1.25)^2 + (1 - 1.25)^2 + (3 - 1.25)^2 + (0 - 1.25)^2]/4}$.

Based on this definition, we consider than an itemset X is periodic in a sequence s if it meets the following conditions.

Definition 4 (periodic pattern in a sequence). Let there be three user-specified thresholds $maxPer, minSup$ and $maxStd$. An itemset X is periodic in a sequence s if $maxPr(X, s) \leq maxPr, sup(X, s) \geq minSup$ and $stanDev(X, s) \leq maxStd$.

For example, the itemset $\{a, b\}$ is periodic in sequence s_1 for $maxStd = 2.0$, since $stanDev(\{a, b\}, s_1) = 1.09$. The above definition is proposed to identify periodic patterns in a single sequence. The following paragraphs explain how it is extended to discover periodic frequent patterns common to multiple sequences using a novel measure called the sequence periodic ratio.

Definition 5 (sequence periodic ratio). The number of sequences where an itemset X is periodic in a sequence database D is denoted as $numSeq(X)$. The sequence periodic ratio of X in D is defined as $ra(X) = numSeq(X)/|D|$, where $|D|$ is the number of sequences in D .

For instance, the number of sequences where $\{a, b\}$ is periodic is $numSeq(\{a, b\}) = 3$ (it is periodic in s_1, s_2 , and s_4). The total number of sequences is $|D| = 4$. Thus, the sequence periodic ratio of $\{a, b\}$ is $ra(\{a, b\}) = 3/4 = 0.75$.

Problem Statement. Let there be a sequence database D , and four user-defined thresholds, namely the minimum support threshold $minSup$, maximum periodicity threshold $maxPr$, maximum standard deviation threshold $maxStd$, and minimum sequence periodic ratio threshold $minRa$. An itemset X is a Periodic Frequent Pattern (PFPS) in D if $ra(X) \geq minRa$. The problem of mining periodic patterns common to multiple sequences is to find all PFPS.

For example, Table 2 shows the PFPS found for different thresholds values. The first line uses $minSup = 2, maxPr = 3, maxStd = 5.0$ and $minRa = 0.3$, while the following lines change one parameter with respect to the first line (in bold). It can be seen that each parameter is useful to reduce the number of patterns, and thus provide a lot of flexibility to the user to select the desired patterns.

Also, we introduce a new measure called $boundRa$ that is an upper-bound on the ra measure to be able to reduce the search space.

Table 2. Patterns found for different threshold values

No.	<i>minSup</i>	<i>maxPr</i>	<i>maxStd</i>	<i>minRa</i>	Patterns found
1	2	3	1.0	0.6	$\{a\}, \{e\}, \{a, e\}$
2	3	3	1.0	0.6	$\{a\}$
3	2	1	1.0	0.6	$\{a\}$
4	2	3	1.5	0.6	$\{a\}, \{b\}, \{e\}, \{a, b\}, \{a, e\},$
5	2	3	1.0	0.4	$\{a\}, \{b\}, \{c\}, \{e\}, \{a, b\}, \{a, e\}, \{b, e\}, \{a, b, e\}$

Definition 6 (boundRa). Given two user-specified thresholds $maxPr$ and $minSup$, an itemset X is a candidate in a sequence s if $maxPr(X, s) \leq maxPr$ and $sup(X, s) \geq minSup$. The number of sequences where an itemset X is a candidate in a sequence database D is denoted as $numCand(X)$. The $boundRa$ of X in D is defined as $boundRa(X) = numCand(X)/|D|$.

Property 1 (pruning property). For two itemsets $X \subseteq X'$, (1) $boundRa(X) \geq ra(X)$ and (2) $boundRa(X) \geq boundRa(X')$.

Proof. It can be easily seen that the relationship (1) holds since the definition of $ra(X)$ and $boundRa(X)$ are the same except that the former adds one more constraint. The relationship (2) is proven as follows. In a sequence s , it was shown

that $\maxPr(X, s) \leq \maxPr(X', s)$ [9], and that $\sup(X, s) \leq \sup(X', s)$ [2]. Thus, the number of sequences where X' is a candidate cannot be greater than the number of sequences where X is a candidate, i.e. $\text{numCand}(X') \leq \text{numCand}(X)$. Hence, $\text{boundRa}(X) \geq \text{boundRa}(X')$. \square

4 The MPFPS Algorithm

This section introduces an efficient algorithm to mine PFPS, named MPFPS (Mining Periodic Frequent Pattern common to Sequences). MPFPS explores the search space of itemsets using a depth-first search. The search space consists of $2^{|I|}$ itemsets. MPFPS starts from single items, and recursively appends an item to each itemset to generate a larger itemset, following the \succ order. MPFPS reduces the search space by exploiting the fact that boundRa is an upper-bound on the ra measure and satisfies the *downward closure property* (Property 1). To calculate all the measures to evaluate patterns without having to repeatedly scan the database, MPFPS utilizes a novel data structure called PFPS-list. A PFPS-list is created for each itemset that is visited in the search space. The PFPS-list of an itemset X contains three fields. The *i-set* field stores X . The *tidlist-list* field stores the list of identifiers of transactions containing X . The *sid-list* field contains the list of identifiers of sequences containing X . For example, the PFPS-list of the itemset $\{a\}$ is: *i-set*: $\{a\}$, *tidlist-list*: $[\{0,1,2,3,4\}, \{1,3,4\}, \{1,2,3,4\}, \{0,1,2,3,4\}]$, *sid-list*: $\{0,1,2,3\}$. The PFPS-list of an itemset allows to quickly find the transactions and sequences where it appears, and thus to determine if the itemset is periodic without scanning the database. Moreover, as it will be explained, the PFPS-list of any itemset containing more than one item can be obtained by performing an intersection operation on the PFPS-lists of two of its subsets (without scanning the database).

The proposed MPFPS (Algorithm 1) takes as input a database with multiple sequences and the maxStd , minRa , maxPr , and minSup thresholds. MPFPS outputs all the PFPS. It first scan the database once to calculate $\sup(\{i\}, s)$, $\text{pr}(\{i\}, s)$, $\text{maxpr}(\{i\}, s)$ and $\text{stanDev}(\{i\}, s)$ for each item i and sequence s (line 1). Then, MPFPS checks if each item i is periodic in each sequence of the database (line 2 to 4). For an item i appearing in a sequence s , if $\sup(\{i\}, s) \geq \text{minSup}$, $\text{maxpr}(\{i\}, s) \leq \text{maxPr}$ and $\text{stanDev}(\{i\}, s) < \text{maxStd}$, then i is said to be periodic in that sequence. Then, the algorithm calculates the sequence periodic ratio of item i by dividing the number of sequences where i is periodic by the total number of sequences. If this value is not less than minRa , i is a PFPS (line 4). Also, boundRa of $\{i\}$ is calculated to prune the search space (line 5 and 6) and the PFPS-list of each single item i such that $\text{boundRa}(\{i\}) \geq \text{minRa}$ is stored in a set boundPFPSSingle (line 8), which is sorted according to the \succ order of ascending support. Then, the depth-first search of PFPS starts by calling the recursive procedure *Search* with boundPFPSSingle , minSup , maxPr , maxStd , minRa and D .

The *Search* procedure (Algorithm 2) takes as input PFPS-lists of extensions of an itemset P , the minSup , maxPr , maxStd , minRa thresholds and

Algorithm 1. The MPFPS algorithm

```

input :  $D$ : a database with multiple sequences,  $maxStd, minRa, maxPr, minSup$ : the
thresholds.
output: the set of periodic frequent patterns (PFPS).
1 Scan each sequence  $s \in D$  to calculate  $sup(\{i\}, s)$ ,  $pr(\{i\}, s)$ ,  $maxpr(\{i\}, s)$  and
 $stanDev(\{i\}, s)$  for each item  $i \in I$ ;
2 foreach item  $i \in I$  do
3    $numSeq(\{i\}) \leftarrow |\{s | maxpr(\{i\}, s) \leq maxPr \wedge stanDev(\{i\}, s) \leq$ 
 $maxStd \wedge sup(\{i\}, s) \geq minSup \wedge s \in D\}|$ ;
4    $ra(\{i\}) \leftarrow numSeq(\{i\})/|D|$ ;
5    $numCand(\{i\}) \leftarrow |\{s | maxpr(\{i\}, s) \leq maxPr \wedge sup(\{i\}, s) \geq minSup \wedge s \in D\}|$ ;
6    $boundRa(\{i\}) \leftarrow numCand(\{i\})/|D|$ ;
7 end
8  $boundPFPSsingle \leftarrow \{PFPS\text{-list of item } i | i \in I \wedge boundRa(\{i\}) \geq minRa\}$ ;
9 Sort  $boundPFPSsingle$  by the order  $\succ$  of ascending support values;
10 Search ( $boundPFPSsingle, minSup, maxPr, maxStd, minRa, D$ );

```

the database. An *extension* of an itemset P is an itemset that is obtained by appending an item z to P , and is denoted as Pz . When the procedure is first called, P is the empty set, and extensions of P are single items. The *Search* procedure performs a loop for each extension Px of P . The procedure first calculates $numCand(Px)$ and $boundRa(Px)$ using the PFPS-list of Px , which is denoted as LPx (line 2 to 3). Then, if $boundRa(Px) \geq minRa$, $ra(Px)$ is calculated to check if Px is a PFPS, and if yes, Px is output (line 5 to 7). Then, a loop is performed to combine Px with each extension P_y of P such that $y \succ x$, to generate an extension Pxy containing $|Px| + 1$ items (line 8 to 11). The PFPS-list of each such extension Pxy , denoted as $LPxy$ is created without scanning the database by applying the *Intersect* procedure (Algorithm 3) on the PFPS-lists of Px and P_y . Then, the *Search* procedure is recursively called with the PFPS-lists of PFPS that extend Px to explore their transitive extensions (line 13).

Algorithm 2. The *Search* procedure

```

input :  $ExtensionsOfP$ : a set of PFPS-lists of extensions of an itemset  $P$ ,
 $minSup, maxPr, maxStd, minRa$ : the thresholds,  $D$ : the database.
output: the set of periodic frequent patterns that extend  $P$ .
1 foreach PFPS-list  $LPx \in ExtensionsOfP$  and  $Px = LPx.i\text{-set}$  do
2    $numCand(Px) \leftarrow |\{s | maxpr(Px, s) \leq maxPr \wedge sup(Px, s) \geq minSup \wedge s \in D\}|$ ;
3    $boundRa(Px) \leftarrow numCand(Px)/|D|$ ;
4   if  $boundRa(Px) \geq minRa$  then
5      $numSeq(Px) \leftarrow |\{s | maxpr(Px, s) \leq maxPr \wedge stanDev(Px, s) \leq$ 
 $maxStd \wedge sup(Px, s) \geq minSup \wedge s \in LPx.sid\text{-list}\}|$ ;
6      $ra(Px) \leftarrow numSeq(Px)/|D|$ ;
7     if  $ra(Px) \geq minRa$  then output  $Px$  ;
8     foreach PFPS-list  $LP_y \in ExtensionsOfP$  and  $P_y = LP_y.i\text{-set}$  such that  $y \succ x$ 
9     do
10       $LPxy \leftarrow Intersect(LPx, LP_y)$ ;
11       $ExtensionsOfPx \leftarrow ExtensionsOfPx \cup \{LPxy\}$ ;
12     end
13   end
14 Search ( $ExtensionsOfPx, minSup, maxPr, maxStd, minRa, D$ );

```

The *Intersect* procedure takes as input the PFPS-lists of two itemsets Px and P_y , denoted as LPx and LP_y , and outputs the PFPS-list of the itemset

Pxy . The algorithm first initialize an empty PFPS-list $LPxy$ for Pxy (line 1). Then, the algorithm performs a loop to consider each sequence that appears in both the PFPS-lists of Px and Py . For each such sequence, let si be the sequence identifier representing that sequence. That sequence identifier is used to retrieve the lists of identifiers of transactions containing Px and Py in that sequence, denoted as $tidListSiPx$ and $tidListSiPy$, respectively (line 3 and 4). These two lists are then intersected to obtain the list of identifiers of transactions containing Pxy in that sequence, named $tidListSiPxy$ (line 5). If that latter list is not empty, si is added to the PFPS-list of Pxy as well as the list of transactions $tidListSiPxy$. The procedure returns the PFPS-list of Pxy , which is obtained without scanning the database.

Algorithm 3. The Intersect procedure

```

input :  $LPx$  and  $LPy$ : the PFPS-lists of two extensions  $Px$  and  $Py$  of an itemset
output: the PFPS-list  $LPxy$  of itemset  $Pxy$ 
1  $LPxy.i\text{-set} \leftarrow Px \cup \{y\}$ ;  $LPxy.tidlist\text{-list} \leftarrow \emptyset$ ;  $LPxy.sid\text{-list} \leftarrow \emptyset$ ;
2 foreach sequence identifier  $si \in LPx.sid\text{-list}$  such that  $si \in LPy.sid\text{-list}$  do
3    $tidListSiPx \leftarrow$  the tid list of  $si$  in  $LPx.tidlist\text{-list}$ ;
4    $tidListSiPy \leftarrow$  the tid list of  $si$  in  $LPy.tidlist\text{-list}$ ;
5    $tidListSiPxy \leftarrow tidListSiPx \cap tidListSiPy$ ;
6   if  $tidListSiPxy \neq \emptyset$  then
7      $LPxy.sid\text{-list.append}(si)$ ;  $LPxy.tidlist\text{-list.append}(tidListSiPxy)$ ;
8   end
9 end
10 return  $LPxy$ ;
```

4.1 A Detailed Example

Consider the example database of Table 1 and that $minSup = 2, maxPr = 3, maxStd = 1.0$ and $minRa = 0.6$. The main procedure of MPFPS (Algorithm 1) first processes single items. Consider the item a . By scanning the database, it finds that $pr(\{a\}, s_1) = [1,1,1,1,1,0]$, $pr(\{a\}, s_2) = [2,2,1,0]$, $pr(\{a\}, s_3) = [2,1,1,1,0]$, $pr(\{a\}, s_4) = [1,1,1,1,1,0]$, $maxpr(\{a\}, s_1) = 1$, $maxpr(\{a\}, s_2) = 2$, $maxpr(\{a\}, s_3) = 2$, $maxpr(\{a\}, s_4) = 1$, $stanDev(\{a\}, s_1) = 0.152$, $stanDev(\{a\}, s_2) = 0.256$, $stanDev(\{a\}, s_3) = 0.632$, and $stanDev(\{a\}, s_4) = 0.152$. As $maxpr(\{a\}, s_1) < maxPr$ and $stanDev(\{a\}, s_1) < maxStd$, item $\{a\}$ is periodic in sequence s_1 . Similarly, it is also periodic in sequences s_2, s_3 and s_4 , and $\{a\}$ is periodic in $numSeq(\{a\}) = 4$ sequences and the ratio $ra(\{a\}) = 1 \geq minRa$. Thus, $\{a\}$ is a PFPS. Then, the same process is repeated for the other items and it is found that the items $\{a\}$ and $\{e\}$ are PFPS. The PFPS-lists of these items are built, and the *Search* procedure is called to find larger PFPS (Algorithm 2). Since $\{a\}$ is a PFPS and $boundRa(\{a\}) \geq minRa$, the *Search* procedure outputs a . Then, the loop is continued with the PFPS-lists of extensions of \emptyset , that is the PFPS-list of $\{e\}$. The Intersect procedure is applied on the PFPS-lists of $\{a\}$ and $\{e\}$ to generate the PFPS-list of $\{a, e\}$. Then, the sequences in which $\{a\}$ and $\{e\}$ are both periodic and candidates are found. The *sid-list* of $\{a\}$ is $\{0, 1, 2, 3\}$ and that of $\{e\}$ is *sid-list* is $\{0, 1, 3\}$. Thus, they are both periodic in the first, second and fourth sequences. The *sid-list* of

$\{a, e\}$ is $\{0, 1, 3\}$. For the first sequence, $\{a\}$'s tidlist is $\{0, 1, 2, 3, 4\}$, $\{e\}$'s tidlist is $\{0, 1, 3\}$, and their intersection is $\{0, 1, 3\}$. Hence, itemset $\{a, e\}$ is periodic in this sequence. The sequence identifier 0 is added to the *sid-list* of $\{a, e\}$ and the intersection $\{0, 1, 3\}$ to the *tidlist-list* of $\{a, e\}$. The same process is repeated for the two other sequences, and the PFPS-list of $\{a, e\}$ is returned to the *Search* procedure, which adds it to the PFPS-list extensions of $\{a\}$ and it recursively calls itself until all PFPS are found.

5 Experimental Evaluation

In prior work, a single algorithm named PHUSPM was proposed to mine periodic patterns in a sequence database. However, it finds patterns that regularly appear in a database whereas the proposed MPFPS algorithm finds patterns that are periodic in many sequences. Thus, the performance of these algorithms cannot be compared. For this reason, this experimental evaluation compares the performance of MPFPS for several parameter values with a baseline version of MPFPS designed to find all frequent patterns. MPFPS is implemented in Java, and was run on a Windows 10 computer equipped with a 3.60 GHz Xeon E3 CPU with 64 GB of memory. The experiment is carried out on two real databases (*FIFA*, *Bible*) obtained from the website of the SPMF library [7], and on a synthetic sequence database created using the SPMF generator. These databases were chosen because they have different characteristics. *FIFA* contains 2,990 distinct items and 20,450 sequences, with an average of 34.7 items per sequence. *Bible* contains 13,905 distinct items and 36,369 sequences, with an average of 21.6 items per sequence. In *FIFA* and *Bible*, each transaction contains one item per transaction. For this reason, the synthetic *T15I1KD300K* database was also used, where each transaction has at most 10 items per transactions. This large sparse sequence database contains 300,000 transactions and 1,000 distinct items, and 10 items on average per sequence. The source code of MPFPS and datasets can be obtained from <http://www.philippe-fournier-viger.com/spmf/>.

MPFPS has four parameters: *maxStd*, *minRa*, *maxPr* and *minSup*. The latter is used to find frequent patterns in each sequence and has been found to have little influence on the number of periodic patterns. Hence, *minSup* is set to a fixed value in the experiment (*minsup* = 2). Moreover, when *maxStd* and *maxPr* are set to large values and *minRa* = 0, the algorithm finds all frequent patterns. Thus, in the following, the MPFPS algorithm with *maxStd* = 1000, *minRa* = 0 and *maxPr* = 1000 will be used as baseline, and will be denoted as *MPFPS-baseline*(0, 1000). The baseline will be compared with other parameter settings to assess the influence of the *maxStd*, *maxPr* and *minRa* parameters on the performance of the algorithm and number of patterns found. In the following, *MPFPS*(*x*, *y*) denotes MPFPS with *minRa* = *x*, *maxPr* = *y* and *minSup* = 2.

5.1 Influence of *minRa* and *maxStd*

The *minRa* and *maxStd* parameters were first varied to evaluate their joint influence, while the *maxPr* parameter was set to a fixed value (*maxPr* = 10

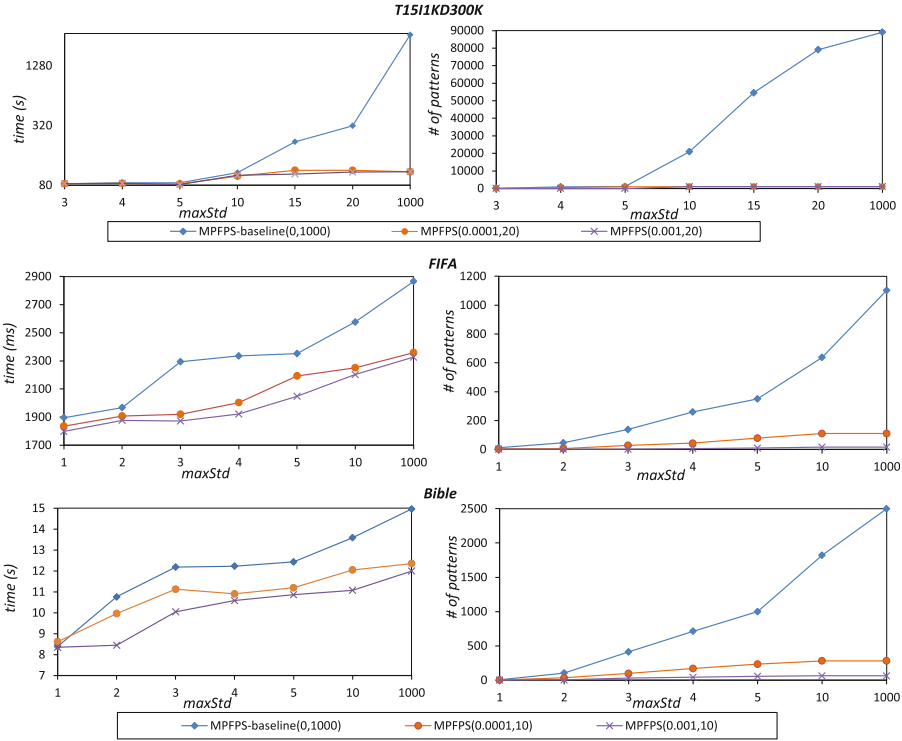


Fig. 1. Runtimes and number of patterns for different $minRa$ and $maxStd$ values

$maxStd = 1000$, $minRa = 0$ and $maxPr = 5$, 134 patterns are found, while 1000 patterns are found for $maxPr = 1000$. This is reasonable since the condition that the periods of a pattern must all be less than $maxPr$ is very strict.

In terms of runtime, decreasing the $maxPr$ parameter improves performance but not by a large amount on the real *FIFA* and *Bible* datasets, while it provides a considerable improvement on the synthetic dataset. The reason is that the search space is larger in that dataset and pruning properties are more effective as few periodic patterns are expected to appear in this database since the data is synthetic.

5.3 Memory Used for Different Parameter Values

Tables 3, 4 and 5 compares the memory used for various values of $minRa$ and $maxPr$, when $maxStd$ is varied as in the previous subsection for the three databases. Due to the space limitation, for the *FIFA* and *Bible* databases, only three lines of results are shown. The omitted lines are almost the same (around 320 MB).

It can firstly be observed that, when $maxStd$ is increased, MPFPS needs more space to store the PFPS-lists of the itemsets. For example, when $maxStd$

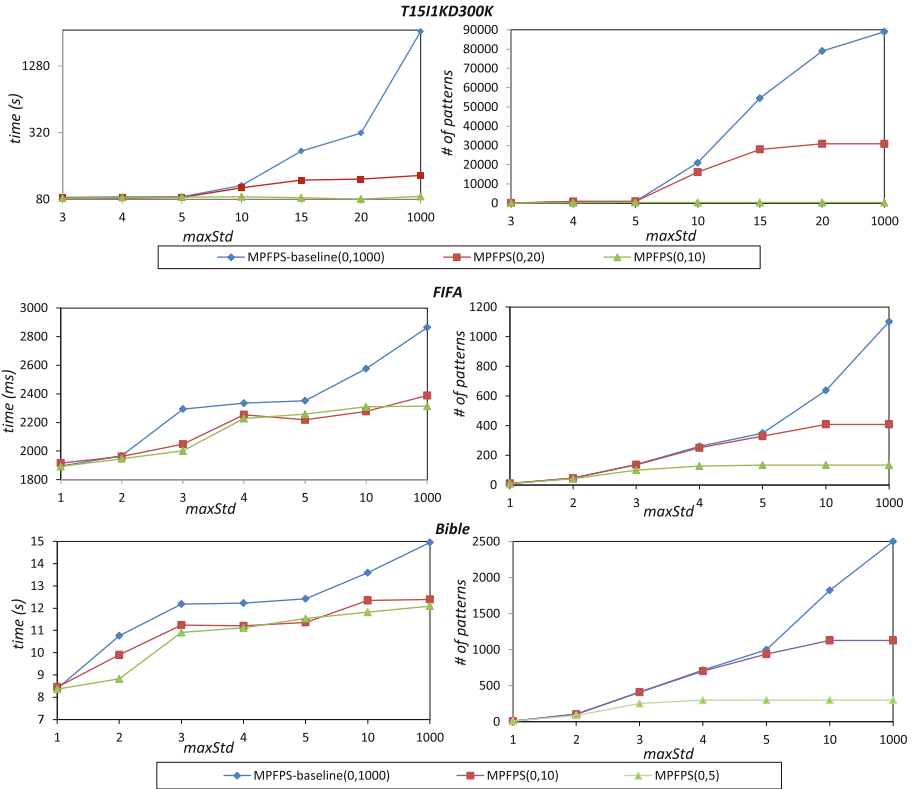


Fig. 2. Runtimes and number of patterns for different *maxPr* and *maxStd* values

is increased from 3 to 15 for MPFPS(0.001, 20) in Table 3, memory usage also increases from 1797 MB to 2589 MB. This shows that *maxStd* can be used to greatly reduce the number of patterns stored in memory.

For the *minRa* parameter, it is observed that this parameter also influences memory usage. For example, for *T15I1KD300K* when *minRa* is increased from 0 to 0.001, *maxStd* = 4 and *maxPr* = 20, the memory decreases from 1995 MB to 1797 MB. This shows that *minRa* is also helpful for reducing the number of stored patterns.

The *maxPr* threshold can also reduce memory usage, as observed in the tables. For example, for *T15I1KD300K*, when *maxPr* is decreased from 1000 to 10, *maxStd* = 10 and *minRa* = 0, the memory decreases from 2452 MB to 1837 MB.

5.4 Discussion

On overall, it has been found that the *maxPr*, *minRa* and *maxPr* parameters can be used to filter a large number of non periodic patterns. Thus the proposed

Table 4. Memory used by MPFPS for different parameter values on *FIFA*

Memory(MB) \ <i>maxStd</i>	1	2	3	4	5	10	1000
MPFPS(<i>x,y</i>)							
MPFPS-baseline(0,1000)	280	320	320	320	320	400	525
MPFPS(0,10)	280	320	320	320	360	360	360
MPFPS(0.0001,10)	320	320	320	320	320	320	320

Table 5. Memory used by MPFPS for different parameter values on *Bible*

Memory(MB) \ <i>maxStd</i>	1	2	3	4	5	10	1000
MPFPS(<i>x,y</i>)							
MPFPS-baseline(0,1000)	280	320	380	400	520	847	1386
MPFPS(0,10)	280	320	360	400	480	560	560
MPFPS(0.0001,10)	280	320	320	320	320	320	320

algorithm can be used to find a small set of periodic patterns. It was also shown that parameters can also reduce the runtime. How to set the parameters is dataset dependent as different databases may contain patterns with shorter or longer periods, or periods that vary more or less greatly. It has been found that the *minSup* parameter has a very small influence on the algorithm’s output and performance (results were not shown due to space limitation). Thus, it is recommended to set the *minSup* parameter to a small value. It is recommended to set the *maxPr* parameter to a large value as the pruning condition for this parameter is very strict (as explained in the related work section). It should thus be used to filter patterns that have very large periods. Thus, the two most important parameters are the *maxStd* and *maxRa* parameters. The former allows to specify the maximum variation in terms of periodicity over time for a periodic patterns. The latter is the ratio of sequences where a pattern must be periodic, and is proposed to find periodic patterns common to multiple sequences.

6 Conclusion

Previous work on periodic pattern mining have mainly focused on analyzing patterns in a single sequence. This paper has thus proposed a novel problem of mining periodic frequent patterns common to multiple sequences, which consists of discovering all patterns respecting user-defined minimum support, maximum periodicity, maximum standard deviation and minimum sequence periodic ratio thresholds. The problem was formally defined and properties of the problem have been studied. Moreover, an efficient algorithm named MPFPS was proposed to discover these patterns, based on a novel PFPS-list structure and *boundRa* upper-bound to reduce the search space. Experiments on several databases have

shown that the proposed algorithm is effective and can greatly reduce the number of patterns found by filtering non periodic patterns.

There are several opportunities for future work. In future work, we will consider adapting the proposed model for various distributions. Furthermore, we will consider extending the model to discover more complex types of periodic patterns such as partial orders, rules and sequential patterns from sequences.

References

1. Agrawal, R., Srikant, R.: Mining sequential patterns. In: 11th International Conference on Data Engineering, pp. 3–14. IEEE Press, Taipei (1995)
2. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: 20th International Conference on Very Large Data Bases, pp. 487–499. Morgan Kaufmann, Santiago de Chile (1994)
3. Aggarwal, C.C., Han, J.: *Frequent Pattern Mining*. Springer, Switzerland (2014). <https://doi.org/10.1007/978-3-319-07821-2>
4. Zaki, M.J., Gouda, K.: Fast vertical mining using difffsets. In: 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 326–335. ACM, Washington (2003)
5. Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., Hsu, M.: PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth. In: 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 355–359. ACM, Boston (2000)
6. Surana, A., Kiran, R.U., Reddy, P.K.: An efficient approach to mine periodic-frequent patterns in transactional databases. In: Cao, L., Huang, J.Z., Bailey, J., Koh, Y.S., Luo, J. (eds.) PAKDD 2011. LNCS (LNAI), vol. 7104, pp. 254–266. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28320-8_22
7. Fournier-Viger, P., Gomariz, A., Gueniche, T., Soltani, A., Wu, C., Tseng, V.S.: SPMF: a Java open-source pattern mining library. *J. Mach. Learn. Res.* **15**(1), 3389–3393 (2014)
8. Fournier-Viger, P., Lin, J.C.-W., Duong, Q.-H., Dam, T.-L.: PHM: mining periodic high-utility itemsets. In: Perner, P. (ed.) ICDM 2016. LNCS (LNAI), vol. 9728, pp. 64–79. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41561-1_6
9. Fournier-Viger, P., Lin, C.-W., Duong, Q.-H., Dam, T.-L., Sevcic, L., Uhrin, D., Voznak, M.: PFFPM: discovering Periodic Frequent Patterns with Novel Periodicity Measures. In: 2nd Czech-China Scientific Conference, pp. 1–13 (2017)
10. Fournier-Viger, P., Lin, J.C.-W., Vo, B, Chi, T.T., Zhang, J., Le, H.B.: A survey of itemset mining. *J. WIREs Data Mining Knowl. Discov.* **7**(4) (2017)
11. Kiran, R.U., Reddy, P.K.: Mining rare periodic-frequent patterns using multiple minimum supports. In: 15th International Conference on Management of Data, pp. 7–8. IEEE, Mysore (2009)
12. Kiran, R.U., Kitsuregawa, M., Reddy, P.K.: Efficient discovery of periodic-frequent patterns in very large databases. *J. Syst. Softw.* **112**, 110–121 (2016)
13. Amphawan, K., Lenca, P., Surarerks, A.: Mining Top-*K* periodic-frequent pattern from transactional databases without support threshold. In: Papasratorn, B., Chutimaskul, W., Porkaew, K., Vanijja, V. (eds.) IAIT 2009. CCIS, vol. 55, pp. 18–29. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10392-6_3
14. Fan, Y., Ye, Y., Chen, L.: Malicious sequential pattern mining for automatic malware detection. *Expert Syst. Appl.* **52**, 16–25 (2016)

15. Halder, S., Samiullah, M., Lee, Y.-K.: Supergraph based periodic pattern mining in dynamic social networks. *Expert Syst. Appl.* **72**, 430–442 (2017)
16. Dinh, T., Huynh, V.-N., Le, B.: Mining periodic high utility sequential patterns. In: Nguyen, N.T., Tojo, S., Nguyen, L.M., Trawiński, B. (eds.) *ACIIDS 2017. LNCS (LNAI)*, vol. 10191, pp. 545–555. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54472-4_51



Discovering Tight Space-Time Sequences

Riccardo Campisano¹, Heraldo Borges¹, Fabio Porto², Fabio Perosi³,
Esther Pacitti⁴, Florent Masegla⁴, and Eduardo Ogasawara¹✉

- ¹ CEFET/RJ - Federal Center for Technological Education of Rio de Janeiro,
Rio de Janeiro, Brazil
eogasawara@ieee.org
- ² LNCC - National Laboratory of Scientific Computing, Petrópolis, Brazil
- ³ UFRJ - Federal University of Rio de Janeiro, Rio de Janeiro, Brazil
- ⁴ Inria and University of Montpellier, Montpellier, France

Abstract. The problem of discovering spatiotemporal sequential patterns affects a broad range of applications. Many initiatives find sequences constrained by space and time. This paper addresses an appealing new challenge for this domain: find tight space-time sequences, *i.e.*, find within the same process: (i) frequent sequences constrained in space and time that may not be frequent in the entire dataset and (ii) the time interval and space range where these sequences are frequent. The discovery of such patterns along with their constraints may lead to extract valuable knowledge that can remain hidden using traditional methods since their support is extremely low over the entire dataset. We introduce a new *Spatio-Temporal Sequence Miner (STSM)* algorithm to discover tight space-time sequences. We evaluate *STSM* using a proof of concept use case. When compared with general spatial-time sequence mining algorithms (*GSTSM*), *STSM* allows for new insights by detecting maximal space-time areas where each pattern is frequent. To the best of our knowledge, this is the first solution to tackle the problem of identifying tight space-time sequences.

1 Introduction

Space and time are pervasive in our day-to-day lives. As many datasets that include both time and space data are becoming available, new opportunities to discover interesting spatiotemporal patterns arise. An event may be classified as an occurrence of a phenomenon in a given space and time. A spatiotemporal sequential pattern is a sequence of events that are constrained in space and time [7]. Due to that, spatiotemporal sequence mining is gaining attention [11, 12].

In this work, we investigate a new problem related to spatiotemporal pattern identification. We are interested in finding tight space-time sequences, *i.e.*, sequences that are constrained in space and time that may not be frequent in the entire dataset but are frequent inside a time interval and space range (spatiotemporal blocks). The primary challenge is to discover these blocks and the frequent sequences they contain. Solving this problem has a valuable impact on many applications.

This paper introduces the problem of discovering tight space-time sequences and proposes an effective and efficient solution with the *Spatio-Temporal Sequence Miner (STSM)* algorithm to address it. *STSM* applies a sequential pattern mining algorithm to detect spatial ranges where sequences are frequent. Next, it composes all detected sequences inside each spatial range to discover time intervals in which these sequences are frequent.

Our group first experienced the spatiotemporal sequence mine problem while analyzing seismic surveys [5]. By applying constrained space-time sequence mining techniques, we managed to identify 1,500 tight space-time sequences under high support threshold, which would not have been found using general spatial-time sequence mining algorithms (*GSTSM*) [12] for the same frequency threshold. Moreover, the algorithm was able to detect candidate areas for seismic *horizons* (which stand for kind of layers) and *bright spots* (potential areas of hydrocarbon accumulation), which are known as important characteristics for the domain experts [15].

In addition to this introduction, this paper is organized into five more sections. Section 2 discusses related works. Section 3 presents the definitions needed to describe our algorithm. Section 4 gives the *STSM* algorithm to identify sequential pattern constrained in space and time. Section 5 details the experimental evaluation. Finally, Sect. 6 concludes the paper.

2 Related Works

Sequential pattern mining is a broad field of research embracing several approaches [11]. Over the last decades, some works have been developed for sequential mining of spatiotemporal datasets [12]. We have conducted a systematic map study querying Scopus using the keywords (“*sequence mining*” \vee “*sequential pattern*”) \wedge (“*space-time*” \vee “*spatiotemporal*”). We identified 98 articles that were related to spatiotemporal sequential pattern mining, which are grouped according to different types of datasets: (i) trajectory datasets, (ii) event-based datasets, and (iii) emerging patterns datasets.

Trajectories datasets can be classified as a collection of events of the same moving object at different spatial locations and times. The goal is to retrieve similar trajectories that might reveal underlying traveling patterns of moving objects in the dataset [7]. It includes works that search for routes frequently used [6]. Since commonly the position of moving objects may be inaccurate, some approaches address the position uncertainty to better recognize common trajectories [10]. Although trajectory problems are relevant, they differ from our problem. We are not interested in moving objects that have the same behavior. Instead, we are interested in regions and time intervals when some events are related (constrained) and relevant (with high support).

Event-based datasets correspond to the majority of related work. The goal is to find sequences constrained by space and time. For that, data is partitioned according to spatial or temporal dimensions, and events are related whenever they preserve certain proximity [14]. Huang et al. [7] define sequence index

as the significance measure for evaluating identified spatiotemporal sequential patterns. Such approach improves the mining algorithm to identify constrained sequences. Julea et al. [8] studied satellite images and target patterns with a high connectivity measure. Alatrasta-Salas et al. [1] expand the search-space for identifying sequences by looking at neighbor partitions. Li et al. [9] address the problem of event prediction from identified spatiotemporal sequence patterns. Finally, Batu et al. [3] avoid partitioning space and time by computing the most relevant sequences in the dataset. Although approaches for event-based datasets are in the same category of our paper, they differ since all identified sequences are frequent in the entire dataset. In our work, sequences may only be frequent inside spatiotemporal blocks (*i.e.*, a time interval and space range).

Finally, emerging pattern datasets correspond to solve the problem where data are continuously added to the database. Consider the patterns identified in the initial dataset and the patterns that are identified in an updated dataset. Previously identified patterns may become irrelevant, and new patterns may emerge. Some initiatives in emerging spatiotemporal datasets have been developed so far [2, 4, 13]. These problems are complementary to ours since all identified patterns in both datasets (initial or updated) have high support. The problem tackled in this paper may find tight space-time sequence in both datasets, which may emerge or not.

Considering the aforementioned related works, to the best of our knowledge, we are not aware of similar works that studied tight space-time sequences from spatiotemporal datasets.

3 Formalization

Let $t = \langle v_1, v_2, \dots, v_n \rangle$ be a **time-stamped sequence (TS)**, *i.e.*, a **sequence** of items, where $|t| = n$ is the number of items in t . A time index j is an integer value between 1 and n that is related to item v_j . A sequence $s = \langle w_1, w_2, \dots, w_k \rangle$ is **included** in another sequence $t = \langle v_1, v_2, \dots, v_n \rangle$ if there exist integers $i_1 < i_2 < \dots < i_k$ such that $w_1 = v_{i_1}, w_2 = v_{i_2}, \dots, w_k = v_{i_k}$. Let $P = \{p_1, p_2, \dots, p_m\}$ be a set of positions, a **spatial time-stamped sequence (STS)** d is a couple (p, t) where $p \in P$ is a position and t is the associated TS. A **STS dataset** D is a set of STS.

An STS $d = (p, t)$ is said to support a sequence s if s is included in t . The **support** of a sequence s in D is the number of STS in D in which s is included. The **frequency** of a sequence s in D is the fraction of STS in D that supports s : $freq(s, D) = \frac{sup(s, D)}{|D|}$. Given a user's minimum threshold $\gamma \in]0..1]$, a sequence s is said to be **frequent** if $freq(s, D) \geq \gamma$. Let I be the set of all possible sequences s included in D , the **set F of frequent sequences** in D concerning γ is denoted by $F(D, \gamma) = \{s \in I : freq(s, D) \geq \gamma\}$.

A **spatial range** (or simply **range**) $r = (p_s, p_e)$ is defined by a start position p_s and an end position p_e . We define the set of all potential ranges over D as PR . The set of STS that belong to range r is defined as $Tr(r) = \{d : d \subseteq D, p_s \leq d.p \leq p_e\}$. The frequency of s over $Tr(r)$ in STS of a range r is denoted

by $freq(s, r)$ whenever it is clear from the context. The same approach is valid for support of s over $Tr(r)$, which is denoted by $sup(s, r)$. A **ranged sequence** sr is a triple (s, r, fr) where s is a *sequence*, r is a *range*, and fr is the *frequency* of the *sequence* s over the *range* r : $fr = freq(s, r)$. Let k be the size of s ; then s is called a k -sequence.

A **time interval** (or simply **interval**) $i = (i_s, i_e)$ is defined by a start time i_s and an end time i_e . The length of an interval i is given by: $|i| = i_e - i_s + 1$. Given an interval i , a sequence $s = \langle w_1, w_2, \dots, w_k \rangle$ is a **subsequence** of another sequence $t = \langle v_1, v_2, \dots, v_n \rangle$: $s = subseq(t, i)$ iff $i_s \geq 1 \wedge i_e \leq n$, $|i| = k$ and $\forall j \in [1..k], w_j = v_{i_s+j-1}$. By definition, s is also included in t . We define the set of all possible intervals over D as PI . A **block** b is a couple (r, i) where r is a range ($r \in PR$) and i is an interval ($i \in PI$). The size of a block b is the product of range size with interval length: $|b| = |b.r| \cdot |b.i|$. We define the set of all possible blocks over a range r as $PB(r)$. Moreover, a **bounding box** between two blocks b_i, b_j : $bb(b_i, b_j)$ is the minimum block b that contains them.

The **occurrences** of a *sequence* s in a block b refer to the positions in which all instances of s is present in b . Thus, $occur(s, b)$ is a set of blocks o , such that $\forall b_j \in o, b_j \subseteq b \mid |Tr(b_j, r)| = 1 \wedge \forall d \in Tr(b_j, r), subseq(d, b_j, i) = s$. The **support** of a *sequence* s in a block b , $sup(s, b)$ is the number of occurrences of s in b : $|occur(s, b)|$. The **item-support** of a *sequence* s in a block b , is the product of support of a *sequence* s in a block b with the length of the *sequence* s : $isup(s, b) = sup(s, b) \cdot |s|$. The **item-frequency** of a *sequence* s in a block b is the fraction of item-support over the size of b : $ifreq(s, b) = \frac{isup(s, b)}{|b|}$. Given a user's minimum threshold $\delta \in]0..1]$, a *sequence* s is said to be **item-frequent** in a block b if $ifreq(s, b) \geq \delta$. A **blocked sequence** sb is a triple (s, b, ifr) where s is a *sequence*, b is a *block*, and ifr is the *item-frequency* of s over b : $ifr = ifreq(s, b)$.

4 STSM

This section is devoted to the presentation of *STSM* (Spatio-Temporal Sequence Miner), our algorithm designed for the identification of solid-blocked sequences in the spatiotemporal dataset. The notion of kernels (range-kernels and block-kernels) introduced in this section allows for extracting solid-blocked sequences efficiently. First, we give an overview of our main algorithm in Sect. 4.2. Later, each relevant function is described in the following sections.

4.1 STS Definitions

Given γ and δ , user's minimum thresholds, respectively, for frequency and item-frequency, we introduce the characteristics of solid-ranged sequence in Definition 1 and solid-blocked sequence in Definition 2.

Definition 1. Let sr be a ranged sequence of range r , sequence s , and frequency fr . Then, sr is called a **solid-ranged sequence** iff the following conditions

hold: (1) $fr \geq \gamma$; (2) $\forall r_2 \in PR \mid r \subseteq r_2$, we have either $(freq(s, r_2) < \gamma)$ or $(sup(s, r_2) = sup(s, r))$ or both; (3) $\forall r_2 \in PR$ such that $r_2 \subset r$, $sup(s, r_2) < sup(s, r)$.

Let k be the size of s then sr is a **k-solid-ranged sequence**. \mathbf{SR}_k is the set of all k -solid-ranged sequences.

Definition 2. Let sb be a blocked sequence with a block b , sequence s , and item-frequency ifr . Then, sb is called a **solid-blocked sequence** iff the following conditions hold: (1) $\exists sr \in \mathbf{SR}_{|s|} \mid b.r \subseteq sr.r$ and $s = sr.s$; (2) $ifr \geq \delta$; (3) $\forall b_2 \in PB(r) \mid b \subseteq b_2$, we have either $(ifreq(s, b_2) < \delta)$ or $(isup(s, b_2) = isup(s, b))$ or both; (4) $\forall b_2 \in PB(r) \mid b_2 \subset b$, $isup(s, b_2) < isup(s, b)$; (5) $sup(s, b) > 1$.

Let k be the size of s ; then sb is a **k-solid-blocked sequence**. \mathbf{SB}_k is the set of all k -solid-blocked sequences.

4.2 General Principle

Algorithm 1 aims to generate solid-blocked sequences (SB) from size 1 to k . It receives as input an STS dataset D , a solid range threshold γ and a solid block threshold δ . It has three main functions: (i) candidate generation, (ii) identification of solid-ranged sequences, (iii) identification of solid-blocked sequences. The algorithm starts from candidates ranged sequences of size 1. They are built from all distinct items presented in D considering its entire range.

It starts a repeat-until loop that computes all solid-ranged sequences SR_k with a frequency greater than or equal to γ . Then, candidate ranged sequences of size $k+1$ are computed from solid ranges SR_k . Once we get empty candidates ranged sequences of size $k+1$, the loop stops.

Finally, all solid-blocked sequences SB_k with item-frequency greater than or equal to δ are computed from identified solid-ranged sequences.

Algorithm 1. Spatio-Temporal Sequence Miner

```

1: function  $STSM(D, \gamma, \delta)$ 
2:    $C_1 \leftarrow generateCandidates(D, nil)$ 
3:    $k \leftarrow 0$ 
4:   repeat
5:      $k \leftarrow k + 1$ 
6:      $SR_k \leftarrow solidRangedSequences(D, C_k, \gamma)$ 
7:      $C_{k+1} \leftarrow generateCandidates(D, SR_k)$ 
8:   until  $C_{k+1} \neq \emptyset$ 
9:   for  $(i \in \{1 \dots k\})$  do
10:     $SB_i \leftarrow solidBlockedSequences(D, SR_i, \delta)$ 
11:   return  $\{SB_1, \dots, SB_k\}$ 

```

4.3 Toy Example

Figure 1 shows an example of spatiotemporal dataset D to better illustrate the definitions and mechanics of $STSM$ algorithm. Each position p_i is associated with an STS d_i in dataset D . Let us consider a frequency threshold $\gamma = \frac{1}{2}$ given by the user. In this context, the only frequent sequence for D is $\langle a \rangle$ with a frequency of $\frac{1}{2}$ using $GSTSM$.

D t	d ₁	d ₂	d ₃	d ₄	d ₅	d ₆	d ₇	d ₈	d ₉	d ₁₀
v ₁	a	b	c	d	τ	θ	i	g	a	h
v ₂	k	l	m	n	p	q	u	s	t	v
v ₃	w	e	e	x	y	m	a	r	δ	α
v ₄	h	o	o	g	e	ι	ε	i	χ	β
v ₅	i	φ	κ	λ	o	z	v	u	ζ	π
v ₆	u	a	ρ	σ	τ	μ	c	d	f	a

Fig. 1. Dataset D for ten spatial-time series with six observations in each.

This dataset presents other sequences that can be identified by $STSM$ according to our definitions. Consider the ranges $r_1([d_1..d_2])$, $r_2([d_8..d_{10}])$, $r_3([d_2..d_5])$, and $r_4([d_7..d_8])$. Both r_3 and r_4 are, respectively part of solid-ranged sequences $sr_3(\langle e, o \rangle, r_3, \frac{3}{4})$ and $sr_4(\langle i, u \rangle, r_4, 1)$. In fact, considering our generate candidate algorithm, solid-ranged sequence sr_3 is obtained from $\langle e \rangle, r_3, \frac{3}{4}$ and $\langle o \rangle, r_3, \frac{3}{4}$.

Additional, r_1 and r_2 are not solid-ranged sequences. They correspond to range kernels for $\langle a \rangle$ identified by algorithm *solidRangedSequences*. Later, they are merged to build a solid-ranged sequence $\langle a \rangle, [d_1..d_{10}], \frac{1}{2}$. As it is, the sequence $\langle a \rangle$ may not appear to be interesting, but if frequency threshold is increased to $\gamma = \frac{2}{3}$, both r_1 and r_4 become part of a solid-ranged sequence for $\langle a \rangle$, whereas no sequence is identified by general spatial-time sequence mining algorithms ($GSTSM$).

Taking into account the relationship between time and space, intuitively, sequence $\langle a \rangle$ still does not appear to be interesting. In fact, the goal of $STSM$ is to identify tight space-time sequences. Consider blocks $b_1 = ([d_2..d_5], [v_3..v_5])$ and $b_2 = ([d_7..d_8], [v_1..v_5])$. The occurrences of $\langle i, u \rangle$ in b_2 are $\{([d_7], [v_1..v_2]), ([d_8], [v_4..v_5])\}$. Adopting an item-frequency threshold $\delta = \frac{2}{5}$, both blocks are part of solid-blocked sequences $sb_1(\langle e, o \rangle, b_1, \frac{1}{2})$ and $sb_2(\langle i, u \rangle, b_2, \frac{2}{5})$. Additionally, sequence a is not produced by $STSM$ algorithm as a solid-blocked sequence for such threshold.

We can observe that using solid-ranged sequences and solid-blocked sequences definitions together with *STSM* algorithm, it is now possible to find sequences that would not be found using higher frequency thresholds. According to the example, the sequence $\langle e, o \rangle$ is supported by the entire dataset with a frequency of $\frac{3}{10}$ occurs on a particular range (*i.e.* $[d_2..d_5]$) and interval (*i.e.* $[v_3..v_5]$) with a frequency of $\frac{3}{4}$ and item-frequency of $\frac{1}{2}$. Also, it filters unwanted sequences that are not tight in space-time, such as $\langle i, u \rangle$ in block $([d_1], [v_5..v_6])$. Sequence $\langle i, u \rangle$ is only solid-blocked sequence in b_2 .

5 Experimental Evaluation

This section presents the experimental evaluation. Section 5.1 starts presenting the dataset used as a proof of concept. Section 5.2 presents the exploratory analysis of *STSM*. Finally, Sect. 5.3 discusses the behavior of the algorithm *STSM* when compared to *GSTSM*.

5.1 Dataset

The dataset chosen as input data is the inline 401 of the *Netherlands Offshore F3 Block* reflection seismic survey [5]. It is composed of 651 *inlines*. The *inline* contains 951 spatiotemporal series of timely-ordered observations collected in different positions on the surface. Each of spatiotemporal series contains 462 observations, a sample of every four milliseconds, representing the different depths of the subsurface.

Seismic surveys are commonly collected by producing perturbations (shots) on the ground or in the ocean. Such shots generate elastic waves that are propagated in the Earth interior. They are reflected and refracted in each material layer and are returned to the surface in a given position and time. A set of receivers positioned on the surface (onshore or offshore) at fixed aligned intervals records the waveform and wave arrival times. The collected wave amplitudes and frequencies are related to the reflection location, and the wave traveled distance. Thus, the early received signals correspond to upper subsurface layers, and vice-versa. As it can be depicted in Fig. 2, for example, each spatiotemporal series can be interpreted as vertical columns of pixels, where negative amplitude values correspond to lighter pixels whereas positive amplitude values correspond to darker pixels. The inline was made available¹ using SAX with an alphabet of size 7.

5.2 Exploratory Analysis

The algorithm proposed in this work allows for users to set solid range threshold γ and solid block threshold δ constraints. Finding adequate values for these thresholds depends on the characteristics of input dataset/application. Lower

¹ <https://github.com/eogasawara/stsm>.

values for such constraints can lead to the identification of a large number of non-useful frequent sequences. Conversely, higher values for these thresholds can result in the detection of a small number of frequent sequences that may become too small to be interesting. All code, experimental data, and results are made available (see Footnote 1).

We explored the combination of solid range threshold γ (60%, 70%, 80%, 90%, and 100%) with solid block threshold δ (10%, 20%, 30%, 40%, and 50%). For sake of comparison, consider the identified sequence $\langle a, a, g, g \rangle$. In this scenario, when γ is under 80%, a single solid range for this sequence is detected covering the entire dataset. Values between 80% and 100% divide the dataset into more fine-grained solid ranges. A consequence of having better solid range division is that the computation required for identifying solid-blocked sequences is reduced. Relaxing γ from 80% to 60% led to an increase 18% in computing time without improving the detection of a solid-blocked sequence. Adopting the extreme value of 100% may, however, limit too much the detection of solid block sequences. Figure 2a depicts sequence occurrences for $\gamma = 80\%$.

Similarly, examining the solid block threshold δ , when relaxing it to 10%, many larger solid-blocked sequences are identified with many overlapping among them. Conversely, when increasing the δ threshold to higher values (30%), it is possible to observe a significant decrease in the number of identified solid-blocked sequences, many of them lose their overlapping. In Fig. 2a, δ is set to 20%.

5.3 Discussions

The goal of this paper is to provide a way to discover sequences that are frequent in tight time and space. By applying *STSM*, we managed to identify sequences that would not have been found using traditional spatiotemporal techniques using the same support. We ranked the identified sequences by density. The density of a sequence s was computed by the mean block size of all solid-blocked sequences for s . A representative high-ranked sequence ($\langle e, e, f \rangle$) is depicted in Fig. 2b.

Despite the simplicity of used ranking and the absence of any significant seismic concept in its logic, it was good enough to prioritize interesting areas. In the seismic domain, it is of particular importance to identify *horizons*, *i.e.*, zones of major nonconformities between geologic boundary materials, usually associated to different lithologies, which produces variations in the collected reflection values [15]. *STSM* algorithm was able to identify known *horizons*. The majority of high-ranked sequences made available in our site match known seismic *horizons* previously known for this dataset [5]. Additionally, *STSM* was also able to find potential areas of accumulation of hydrocarbon, known as *bright spots*. *Bright spots* are rare patterns that occur when there is an inversion of the wave phase. Such pattern is important not only because gas can be interesting for exploration, but also since it can be near to rocks containing oil. For example, in Fig. 2a, the discovered patterns, evidenced in red, are plotted on the top of yellow marks obtained from the previously known *bright spots* for this dataset [5].

Figure 2a shows that *STSM* algorithm has discovered patterns that cover large parts of known *bright spots*.

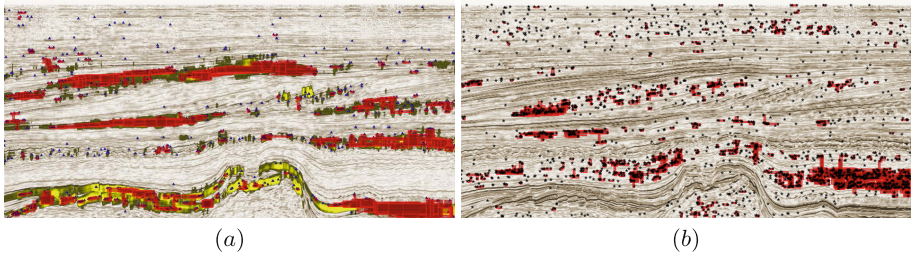


Fig. 2. *inline* 401. Solid-blocked sequence $\langle a, a, g, g \rangle$ are marked in red whereas known *bright spots* are marked in yellow [5] (a) Occurrences of sequence $\langle e, e, d, d \rangle$ from *STSM* red whereas *GSTSM* are black (b) (Color figure online)

Figure 2b shows a comparison between *GSTSM* and *STSM*. The sequence $\langle e, e, f \rangle$ is identified by both algorithms. In *GSTSM* it has high support, whereas in *STSM* it is mid-term ranked results obtained the chosen configuration. It can be observed that although many solid-blocked sequences (depicted in red) are covering the majority of known horizons, many isolated occurrences identified by *GSTSM* (marked as black) correspond to noise. Considering Fig. 2, it is possible to conclude two important differences between *GSTSM* and *STSM*. Not only *STSM* can identify sequences that *GSTSM* is not capable of identifying without lowering too much the frequency threshold, but it also can filter-up sequence occurrences regarding space and time. Such approach allows for researchers to focus on tight sequences.

6 Conclusion

This paper presented a challenging problem with high potential impact: spatiotemporal sequential mining that focuses on discovering frequent tight patterns, along with their constraints of frequency in both space and time. Our formal background provides us with the necessary conditions of such discovery. We proposed *STSM*, a novel algorithm that performs an efficient extraction of these patterns and their spatiotemporal constraints. To the best of our knowledge, this is the first study on this topic. Our experiments, using a real-world seismic dataset, highlighted significant insights according to the feedback of specialists in the domain. *STSM* allows for extracting patterns that follow areas of major nonconformities between geologic boundary materials of the subsurface, such as *horizons* and *bright spots*. The existence of these discovered patterns was confirmed comparing them with a previously known survey on this dataset, while their meaning was assessed by domain experts. Owing to its novelty and

the characteristics of the extracted patterns, *STSM* is opening new research for tight spatial-time sequences discovery.

Although we have evaluated *STSM* algorithm using seismic data, it has been conceived to be sufficiently generic for different spatiotemporal datasets (such as IoT domain). The plot of detected solid-blocked sequences according to space and time, as shown in this paper, can aid specialists to analyze identified sequences. Additionally, in scenarios in which a large number of frequent constrained sequences are produced, we can explore specialized ranking functions to prioritize the most interesting ones.

Acknowledgments. This work was partially funded by CAPES, CNPq, FAPERJ, Inria SciDISC, and the European Union's Horizon 2020 - The EU Framework Programme for Research and Innovation 2014–2020, under grant agreement No. 732051.

References

1. Alatrística-Salas, H., Bringay, S., Flouvat, F., Selmaoui-Folcher, N., Teisseire, M.: Spatio-sequential patterns mining: beyond the boundaries. *Intell. Data Anal.* **20**(2), 293–316 (2016)
2. Aydin, B., Angryk, R.: Spatiotemporal event sequence mining from evolving regions. In: *Proceedings - International Conference on Pattern Recognition*, pp. 4172–4177 (2017)
3. Batu, B., Temizel, T., Duzgun, H.: A non-parametric algorithm for discovering triggering patterns of spatio-temporal event types. *IEEE Trans. Knowl. Data Eng.* **29**(12), 2629–2642 (2017)
4. Chen, Y.L., Hu, Y.H.: Constraint-based sequential pattern mining: the consideration of recency and compactness. *Decis. Support Syst.* **42**(2), 1203–1215 (2006)
5. dgbes: Seismic Interpretation Software & Services. Technical report. <https://dgbes.com/> (2018)
6. Giannotti, F., Nanni, M., Pinelli, F., Pedreschi, D.: Trajectory pattern mining. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 330–339 (2007)
7. Huang, Y., Zhang, L., Zhang, P.: A framework for mining sequential patterns from spatio-temporal event data sets. *IEEE Trans. Knowl. Data Eng.* **20**(4), 433–448 (2008)
8. Julea, A., Méger, N., Bolon, P., Rigotti, C., Doin, M.P., Lasserre, C., Trouve, E., Lazarescu, V.: Unsupervised spatiotemporal mining of satellite image time series using grouped frequent sequential patterns. *IEEE Trans. Geosci. Remote Sens.* **49**(4), 1417–1430 (2011)
9. Li, K., Fu, Y.: Prediction of human activity by discovering temporal sequence patterns. *IEEE Trans. Pattern Anal. Mach. Intell.* **36**(8), 1644–1657 (2014)
10. Li, Y., Bailey, J., Kulik, L., Pei, J.: Mining probabilistic frequent spatio-temporal sequential patterns with gap constraints from uncertain databases. In: *Proceedings - IEEE International Conference on Data Mining, ICDM*, pp. 448–457 (2013)
11. Mooney, C., Roddick, J.: Sequential pattern mining - approaches and algorithms. *ACM Comput. Surv.* **45**(2), 19 (2013)
12. Sunitha, G., Rama Mohan Reddy, A.: Mining frequent patterns from spatiotemporal data sets: a survey. *J. Theor. Appl. Inf. Technol.* **68**(2), 265–274 (2014)

13. Tsai, C.Y., Shieh, Y.C.: A change detection method for sequential patterns. *Decis. Support Syst.* **46**(2), 501–511 (2009)
14. Tsoukatos, I.I., Gunopulos, D.: Efficient mining of spatiotemporal patterns. In: Jensen, C.S., Schneider, M., Seeger, B., Tsostras, V.J. (eds.) *SSTD 2001*. LNCS, vol. 2121, pp. 425–442. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-47724-1_22
15. Zhou, H.W.: *Practical Seismic Data Analysis*, 1st edn. Cambridge University Press, New York (2014)

Cloud and Database Systems



CloudDBGuard: Enabling Sorting and Searching on Encrypted Data in NoSQL Cloud Databases

Tim Waage and Lena Wiese^(✉)

Institute of Computer Science, University of Göttingen,
Goldschmidtstraße 7, 37077 Göttingen, Germany
{tim.waage,wiese}@cs.uni-goettingen.de

Abstract. Efficient and secure data management in the cloud is a topic of high relevance. Cloud databases play an important role – in particular, because they can offer various options for processing large amounts of data. In this article we discuss several practical aspects of applying property-preserving encryption (PPE) to data stored in wide column stores – an encryption type that allows for secure sorting and searching on encrypted data. We present the CloudDBGuard framework that comprises implementations of several PPE schemes and takes care of the necessary metadata management. CloudDBGuard also makes progress in terms of query interface standardization by providing a unified API that supports both Apache Cassandra and Apache HBase.

1 Introduction

Wide column stores (WCSs) comprise a certain group of non-relational databases (“NoSQL” databases; see [19] for a comprehensive survey). Because of their flexible and efficient behavior, WCSs are widely used by cloud storage providers. This raises concerns how confidential data can be protected from a curious cloud storage provider or other attacks by unknown third parties. So far, WCSs do not offer any means to ensure the protection of confidential data from unauthorized access. Our main goal was hence to leverage secure and user-adaptable property-preserving encrypted storage of data in this type of databases. Our prior work [18] focused on developing a framework to enable the seamless integration of property-preserving encryption with the WCSs HBase and Cassandra – see [16] for the implementation details. Here we report on various extensions of our framework that improve runtime properties and functionality:

- For some encryption schemes we were able to improve their runtime further.
- Moreover, we were able to hide the complexity of the used encryption strategies behind an easy-to-learn API (application programming interface), which simplifies the practical usage of the framework.
- Furthermore, the user has to keep just one master password in mind instead of having to deal with several cryptographic passwords.

- We report on a combined benchmarking that applies several of these encryption schemes in one query.

The article is organized as follows. Section 2 provides the necessary background on wide column stores, property-preserving encryption and onion layer encryption. Section 3 gives an in-depth description of the components of our framework. Section 4 reports on the runtime experiments with 10 benchmark queries that combine several property-preserving encryption types. Section 5 concludes this article with a discussion and suggestions for future work.

2 Background and Related Work

2.1 Motivating Example

An application scenario for the CloudDBGuard framework is a mail server that stores confidential emails of several users. While the users are relieved from storing all emails locally, they still require management features from the mail server. PPE enables the mail server to manage the emails efficiently while preserving their confidentiality. For example, a user wants to search for emails containing a certain search word – is enabled by searchable encryption. Furthermore, the user requires the server to sort the emails by date in order to retrieve only the most recent ones or the emails from a specified time interval – this is enabled by property-preserving encryption. In order to benchmark this scenario (in Sect. 4), the widely-used Enron corpus [11] served as our data set. The Enron dataset comprises e-mails of 150 employees of a company. Our test queries combine equality tests (to find a sender exactly), range queries (to find emails in a certain timespan) and word search (on the email body).

2.2 The Data Model of Wide Column Stores

Wide column stores (WCSs) are inspired by Google's BigTable architecture [6]. Several open source databases rely on a very similar data model; yet, they differ in implementation details and access methods (like query languages or APIs). Our framework supports Apache Cassandra [12] and Apache HBase [5].

WCSs use tables, rows and columns like traditional relational (SQL-based) databases. However, the fundamental difference is that columns are created for each row independently instead of being predefined by the table structure (that is, database schema). Every row has at least one mandatory column containing its **identifier** (commonly referred to as “row key”). The identifier of a row has to be unique for the whole table and cannot be used by another row. Rows are maintained in lexicographic order by their identifier. WCSs support data **partitioning** in distributed systems. Ranges of the row identifiers serve as units of distribution such that data are partitioned row-wise (that is, horizontally). Due to this range-based partitioning, similar row identifiers are always kept physically close together in the same partition.

To avoid a fixed database schema, WCSs use an internal data format consisting of key-value-pairs. The key part has several components: the so-called keyspace, a table name, a column name and the row identifier. One of these components is a timestamp, enabling the database to maintain an automatic version control, which can be operated in two ways: either by setting a maximum number of versions to keep, or by specifying a “time-to-live” (TTL) after which data items are to be deleted. More formally WCSs can be considered sparse, distributed, multidimensional maps (see [6]) of the form

$$(\textit{keyspace}, \textit{table}, \textit{column}, \textit{row identifier}, \textit{timestamp}) \rightarrow \textit{value}.$$

The WCS data model is hence different from traditional column stores. The internal workings of WCSs are described in detail in Chap. 8 of [19].

2.3 Property-Preserving Encryption

Property-preserving encryption (PPE) retains certain properties of the plaintext (like order of numerical values) on the ciphertext – or it relies on additional index structures on encrypted values (to support efficient search on encrypted data). The types of PPE relevant for this work are deterministic encryption (DET), order-preserving encryption (OPE) and searchable encryption (SE):

- **DET.** The purpose of DET is enabling the database server to check for equality by mapping identical plaintexts to identical ciphertexts.
- **OPE.** The purpose of OPE is enabling a server to learn the relative order of data elements without revealing their exact values. OPE encrypts two elements p_1, p_2 of a domain D in a such way that $p_1 \leq p_2 \Rightarrow Enc(p_1) \leq Enc(p_2)$ for all $p \in D$. Thus, its use cases are sorting and range queries over encrypted data. A lot of OPE schemes have been proposed with different strategies to map a plaintext to a ciphertext domain (see [2, 7, 10, 20]).
- **SE.** The purpose of SE is enabling a server to search over encrypted data without revealing plaintext data. Most SE schemes use indexes (see [8, 9, 15]), which are encrypted in such a way, that only a token (a so-called trapdoor) sent by the querying user allows for comparing the searchword with the ciphertext. There are also schemes, that avoid having an index by embedding the trapdoor in a special format into the ciphertext itself (see [15]).

2.4 Onion Layer Model

Our framework adapts the basic idea of CryptDB’s [13] onion layer model (OLM): each encrypted data item is surrounded by an outer layer consisting of a strong randomized (RND) encryption. The inner layers consist of the property-preserving encryptions of the data item. The outer RND layer is only removed when the inner layer is needed for query answering. While CryptDB supports relational (SQL) databases, our usage of WCSs requires some changes to the original setting. In particular, row identifier columns must be treated differently from all other columns regarding the onion layer design. They must leak the order of values to allow for row sorting (see the discussion in [18]).

3 The CloudDBGuard Framework

In prior work we identified and implemented database-compatible and practically feasible encryption schemes and quantified their performance with various benchmarks. The following subsections introduce a couple of beneficial extensions of the previously developed concepts.

3.1 API

CloudDBGuard aims at executing queries over encrypted data in wide column stores. Yet, it does not follow the approach of a proxy server between the application and database for re-writing queries, decrypting query results, etc (like various other approaches in this field [13,14]). Instead it introduces an application programming interface (API) taking care of these tasks, that is used by the client application. This API has various advantages over the proxy model:

- A third entity besides client and server can be avoided, which results in less computation and network overhead.
- Most proxy approaches make use of the fact that the majority of SQL queries use a well defined (and rather small) subset of SQL commands. In contrast, some NoSQL databases do not even have query languages. Thus, there would be no uniform way for a proxy to manage incoming requests. The API of CloudDBGuard hides the complexity of the databases' native APIs.
- The WCS data model is realized differently by the databases. In our API the differences in these realizations appear unified for the user.
- The user is able to configure the parameters of the used property preserving encryption schemes much more fine-grained and individually.

The client application using the API of CloudDBGuard runs in a trusted environment. For the API to be able to manage its tasks, it has to maintain auxiliary data, namely keys, metadata and (if necessary) indexes on client side. Since this data has to be stored persistently, it is kept outside the application in the client system's file system. The API manages the database connections, data transfer, encrypting and decrypting. Furthermore it keeps track of metadata and key management. Currently, CloudDBGuard utilizes advanced (index-based) encryption schemes, which allow the system to scale better when datasets become large. Partly, they even provide new functionality (like the ability to search for single words without the need of secondary indexes). The database server never sees any decryption keys, hence it is never able to decrypt private data. Thus, any adversaries (even administrators of the cloud services) are not able to gain sensitive information only from read access. There is no need to change database implementations in order to work with CloudDBGuard.

As FamilyGuard uses encryption schemes that potentially use a high number of cryptographic keys, the manual management of these keys is impractical for the user, but since the database server is not allowed to possess them either, they have to be managed and stored on the client side. This is why FamilyGuard uses

a Java Cryptography Extension KeyStore (JCEKS) provided by the Java Cryptography Extension (JCE) for that task. A JCEKS allows storing an arbitrary number of keys, each of which can be accessed using a custom label. The user has to provide only one single password for the client to gain access to all keys.

3.2 Selective Encryption

Selective encryption only encrypts pre-determined sensitive columns (consider a table of employees, where only the salary has to be kept secret, but not the name, department, etc). Selective encryption helps to save computation time when reading from as well as writing to the database, because it reduces the number of required encryption and decryption operations. It also reduces storage space, because it avoids unnecessary indexes. In the CloudDBGuard API, individual columns can be marked as not to be encrypted.

3.3 Separation of Duties

Separation of Duties describes the concept of more than one person being responsible to complete a task. The same principle can be used to take care of privacy issues that property-preserving encryption alone is not able to cover. Consider again the employee example. If the salary is encrypted with OPE, the exact salary of the employees still does not leak, but it can easily be inferred, who earns the most or who earns more money than others. That might be unwanted. If more than one database instance is available and the available databases are managed by individual and independent authorities, CloudDBGuard can further split up the table in order to avoid such conclusions. The salary column can be stored separated from the rest to avoid the leakage of any connection between salary and employee names. The API of CloudDBGuard supports separation of duties during the process of creating a table. Individual columns of a table can be spread across multiple database instances using the following three strategies:

- Random distribution: In this approach a logical table's columns are distributed randomly across the database instances available for the keyspace.
- Round Robin distribution: This approach distributes columns of a logical table in round robin fashion across the available database instances; the data get distributed as evenly as possible at the point of creating a table.
- Custom Distribution: The user actively specifies, which columns have to be stored separately from which other columns. Considering the example above the user could select the sensitive salary column to be stored separately from all other columns. A more in-depth analysis of customized separation of duties as an optimization problem is given in [3,4].

3.4 Table Profiles

The encryption schemes for order-preserving and searchable encryption have their individual strengths and weaknesses. That is why the API of CloudDBGuard provides the option of specifying so-called *table profiles* when creating

tables. A table profile determines which combination of PPE encryption schemes is actually used during data insertion. The API supports three table profiles:

- **Optimized reading:** This profile prioritizes schemes that have advantages for queries that involve mainly reading from the database. Thus it is the best choice for “write-once” databases. The OPE schemes best suited for fast reading are [10,20]. They have the same type of index, which results in equal reading performance. However, [20] is the preferred choice because it also shows good performance in case of a pre-sorted input (see [17]). For the SE onion layer the scheme of [9] is used. It is the fastest scheme for queries that we have implemented – in particular for repeated queries.
- **Optimized writing:** This profile prioritizes schemes that have advantages for queries that involve mainly writing to the database. Thus, it should be used for scenarios, in which writing occurs more often than reading. The OPE scheme best suited for fast writing is [10], as long as presorted inputs are avoided. For the SE onion layer the scheme of [15] is used; it does not maintain indexes and can insert data faster than [9].
- **Storage efficient:** This profile prioritizes storage needs over computation time and selects the schemes that require the least amount of storage for data and indexes, on client side as well as on server side. Thus, OPE onion layers use [2], since it does not require an index at all. For the same reason SE onion layers use [15].

Other custom table profiles can be added easily. Independent of the used profile, every column gets its own instances of the property-preserving encryption schemes they use. That means, encryption scheme indexes are maintained per column, not per table. This allows the separation of duties as described above; querying answering involves only the index data that is actually required.

3.5 Unification of Data Models

The supported databases Cassandra and HBase follow the WCS data model (see Sect. 2.2). Yet, they differ in the way of achieving that. Using different databases with separation of duties therefore requires analyzing their differences. Here we discuss how the CloudDBGuard API compensates them.

- The WCS data model dictates the existence of (at least) one column that stores unique row identifiers per table. Cassandra and HBase have different ways of addressing this column. Cassandra requires assigning a concrete name and data type for it while creating a table. In contrast, HBase does not need any of that information, because its row identifier columns are unnamed and are always of type byte blob. Cassandra hence requires more precise definitions for the row identifier column; defining a name and data type is thus mandatory when creating tables with CloudDBGuard.
- Some WCSs allow the row identifier to consist of multiple parts. For example, it is possible in Cassandra to combine multiple fields to create a row identifier, which is then called a composite key. A composite key always consists of two

parts. The first part is the partition key (for data distribution across servers). The second part is the clustering key (for storing data within a partition). Both parts can again consist of multiple fields. In contrast, HBase does not know the concept compound row identifiers. If the combination of multiple fields is desired, that has to be created manually (by string concatenation) and stored as a single row identifier. HBase’s native Java API provides prefix filters, that can be used to simulate compound keys. The API of CloudDBGuard hides the complexity of using these from the user.

- Apache Cassandra supports collection types: a field in a row cannot only contain a single value, but also a list, set or map. Elements of these subsets can be addressed by traversal (set), specifying an index (list) or a key (map). In contrast, HBase does not support collection types. Instead, an additional *column qualifier* can be used to realize collection types. With the CloudDBGuard API, the user does not need to care about the difference.
- Cassandra offers a variety of data types, whereas HBase stores everything as byte array. CloudDBGuard follows the HBase approach and stores only byte arrays in encrypted columns in Cassandra as well. On the one hand, except for OPE schemes, outcomes of all used encryption schemes are byte arrays anyway. Storing them as such avoids conversions back to their original data type and saves runtime. Only OPE ciphertexts have to be converted to byte arrays, which can be done fast. On the other hand, seeing only byte blobs in the database makes it much harder for an attacker to infer information.

4 Benchmark

Runtime tests of PPE often assess only one encryption scheme in isolation. In contrast, we benchmark the performance of different types of property-preserving encryption schemes (deterministic, order-preserving and searchable encryption) in their combination. All experiments in this section were run on an Intel Core i7-4600U CPU@2.10 GHz, 8 GB RAM, a Samsung PM851 256 GB SSD using Ubuntu 16.04. The PPE schemes were implemented in Java 8, using cryptographic primitives of the Java Cryptography Extension. The source code is available at <https://github.com/dbsec/FamilyGuard/>. From the Enron corpus [11], we parsed a random subset of 10 000 mails (with $1.03 \cdot 10^7$ words) to simulate an average sized mailbox and created one table row per mail. Our queries select the primary key (the mail identifier) of those rows that meet the specified conditions. We select the primary key, as a unique and a relative small field, in order to let the database fully evaluate these conditions. The small size of the column entries enables to compare database performance without side effects. We wrote queries for each type of scheme separately (Q1–Q3), the six possible combinations of two types (Q4–Q9) and the combination of all the three types (Q10). Details about the columns queried can be found in Table 1.

We tested all queries with each of the three table profiles introduced in Sect. 3.4. Figure 1 shows the results. As can be seen, the query times remain within acceptable time spans of less than two seconds for Cassandra (left) and

Table 1. Types of generated queries

Query	Types of schemes	Queried columns	Query	Types of schemes	Queried columns
Q1	DET	receiver	Q6	DET + SE	sender, body
Q2	OPE	timestamp	Q7	OPE + OPE	sender, timestamp
Q3	SE	body	Q8	OPE + SE	body, timestamp
Q4	DET + DET	sender, receiver	Q9	SE + SE	body, subject
Q5	DET + OPE	sender, timestamp	Q10	DET + OPE + SE	sender, timestamp, subject

even less than one second for HBase (right). This reflects the fact that Cassandra is optimized for writing while HBase is optimized for reading. Furthermore, searchable encryption is by far the most expensive type of property preserving encryption, as all queries involving it take the most time. It has a strong impact, especially when the data fields are large (like in Q9, which involves performing SE on mail bodies). In contrast, deterministic and order-preserving encryption have less impact on runtime.

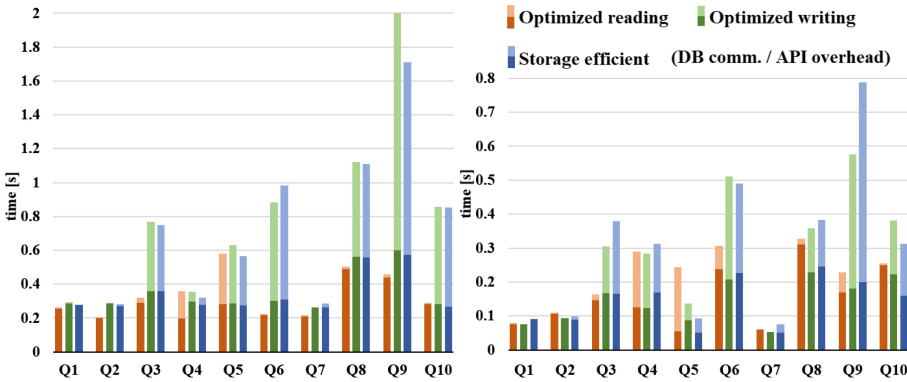


Fig. 1. Query Performance with Cassandra (left) and HBase (right). Measurements for each query: optimized reading (left), optimized writing (middle), storage efficient (right); DB communication (lower part of each bar) + API overhead (upper part)

5 Conclusion and Future Work

We presented the CloudDBGuard framework that extends our prior work in two aspects. Firstly, its functionality was wrapped into an easy-to-use API that

hides the complexity of property-preserving encryption and even the native interface of the underlying database from the user. Secondly, the concept of using property-preserving encryption was combined with other ideas to increase security (e.g., separation of duties) and improve runtime performance (e.g., selective encryption). The framework provides built-in support for Apache Cassandra and HBase. While the concepts of CloudDBGuard are designed for wide column stores, they are not limited to them. Further databases and encryption schemes can be added by implementing simple interfaces. The data model of wide column stores can be mapped easily to key-value stores (in which the key part might be composed as `table:column:qualifier:timestamp`) or document stores (where rows can be mapped to documents and columns can be mapped to fields of a document). The general idea of doing has been suggested by other authors, too (e.g., [1,21]). The major aspect is that the database systems (being open source systems developed by a community) can remain unchanged. Moreover, the API can be integrated into a proxy client between application and database server. In this way, no modifications to the application would be necessary, but an additional architectural component is introduced (as done in the approach of CryptDB [13]). A step further would be the integration of the onion layer model and PPE schemes into the database drivers/native APIs, combining the architectural simplicity of the approach of CloudDBGuard with the opportunity to leave the client application as well as the database server unchanged. This solution however would be very database-specific and the option to transparently use different databases would not be available.

Acknowledgements. This work was funded by the DFG (grant Wi 4086/2-2).

References

1. Atzeni, P., Bugiotti, F., Rossi, L.: SOS (Save Our Systems): a uniform programming interface for non-relational systems. In: Proceedings of the 15th International Conference on Extending Database Technology, pp. 582–585. ACM (2012)
2. Boldyreva, A., Chenette, N., O’Neill, A.: Order-preserving encryption revisited: improved security analysis and alternative solutions. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 578–595. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_33
3. Bollwein, F., Wiese, L.: Closeness constraints for separation of duties in cloud databases as an optimization problem. In: Cali, A., Wood, P., Martin, N., Poulouvasilis, A. (eds.) BICOD 2017. LNCS, vol. 10365, pp. 133–145. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-60795-5_14
4. Bollwein, F., Wiese, L.: Separation of duties for multiple relations in cloud databases as an optimization problem. In: Proceedings of the 21st International Database Engineering & Applications Symposium, pp. 98–107. ACM (2017)
5. Borthakur, D., et al.: Apache Hadoop goes realtime at Facebook. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, pp. 1071–1080. ACM (2011)
6. Chang, F., et al.: Bigtable: a distributed storage system for structured data. *ACM Trans. Comput. Syst.* **26**(2), 4 (2008)

7. Chenette, N., Lewi, K., Weis, S.A., Wu, D.J.: Practical order-revealing encryption with limited leakage. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 474–493. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-52993-5_24
8. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 79–88. ACM (2006)
9. Hahn, F., Kerschbaum, F.: Searchable encryption with secure and efficient updates. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 310–320. ACM (2014)
10. Kerschbaum, F., Schröpfer, A.: Optimal average-complexity ideal-security order-preserving encryption. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 275–286. ACM (2014)
11. Klimt, B., Yang, Y.: The enron corpus: a new dataset for email classification research. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) ECML 2004. LNCS (LNAI), vol. 3201, pp. 217–226. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30115-8_22
12. Lakshman, A., Malik, P.: Cassandra: a decentralized structured storage system. ACM SIGOPS Oper. Syst. Rev. **44**(2), 35–40 (2010)
13. Popa, R.A., Redfield, C.M.S., Zeldovich, N., Balakrishnan, H.: CryptDB: processing queries on an encrypted database. Commun. ACM **55**(9), 103–111 (2012)
14. Popa, R.A., Zeldovich, N., Balakrishnan, H.: Guidelines for using the cryptDB system securely. IACR Cryptology ePrint Archive (2015). <http://eprint.iacr.org/2015/979>, Report 2015/979
15. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: IEEE Symposium on Security and Privacy, pp. 44–55. IEEE (2000)
16. Waage, T.: Familyguard repository. <https://github.com/dbsec/FamilyGuard/>
17. Waage, T., Homann, D., Wiese, L.: Practical application of order-preserving encryption in wide column stores. In: Proceedings of the 13th International Joint Conference on e-Business and Telecommunications - SECURE, pp. 352–359 (2016)
18. Waage, T., Wiese, L.: Property preserving encryption in NoSQL wide column stores. In: Panetto, H., et al. (eds.) OTM 2017. LNCS, vol. 10574, pp. 3–21. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69459-7_1
19. Wiese, L.: Advanced Data Management for SQL, NoSQL, Cloud and Distributed Databases. DeGruyter (2015)
20. Wozniak, S., Rossberg, M., Grau, S., Alshawish, A., Schaefer, G.: Beyond the ideal object: towards disclosure-resilient order-preserving encryption schemes. In: Proceedings of the 2013 ACM Workshop on Cloud Computing Security Workshop, pp. 89–100. ACM (2013)
21. Yuan, X., Wang, X., Wang, C., Qian, C., Lin, J.: Building an encrypted, distributed, and searchable key-value store. In: Proceedings of the 11th ACM Asia Conference on Computer and Communications Security, pp. 547–558. ACM (2016)



Query Processing on Large Graphs: Scalability Through Partitioning

Jay Bodra, Soumyava Das, Abhishek Santra^(✉), and Sharma Chakravarthi

IT Laboratory & CSE Department, UT Arlington, Arlington, USA
{jay.bodra,soumyava.das,abhishek.santra}@mavs.uta.edu,
sharma@cse.uta.edu

Abstract. Graphs, as an expressive data structure, have become increasingly important for modeling real-world applications (collaboration, different kinds of transactions, social networks, to name a few.) With the advent of social networks and the web, the graphs have grown too large to fit in main memory. This calls for alternative approaches, algorithms, and their analysis to develop an efficient, scalable evaluation of queries on graphs of any size.

In this paper, we use the time-tested “divide and conquer” approach by partitioning a graph into desired number of partitions and process queries over those partitions to obtain *all or specified number of answers*. This entails correctly computing answers that span multiple partitions or need the same partition more than once. A query evaluation approach along with the necessary minimal book keeping is proposed and its correctness established. Query answering on partitioned graphs also requires analyzing partitioning schemes for their impact on query processing and determining the number as well as the sequence in which partitions are loaded to reduce the response time for processing one or a batch of queries. We correlate query properties and partition characteristics to reduce query processing time in terms of the number of partitions loaded. We identify a set of quantitative metrics and use them for formulating heuristics to determine the order of loading partitions for efficient query processing. Extensive experiments on large graphs (synthetic and real-world) using different partitioning schemes analyze the proposed heuristics on a variety of query types. An existing graph querying system has been extended to evaluate queries on partitioned graphs.

1 Motivation

Querying transactional data stored in a database or searching of documents/web pages is well-established. Search engines do a very good job of retrieving top-k answers from data repositories. Lately, large data sets are being created (e.g., social networks, web graphs, question/answer graphs, and other ontology-based data sets, to mention a few) that have structural relationships in addition to a wide variety of meta data (multiple labels on nodes and edges, as well as weights on nodes and edges) as part of that structural description. This, in essence,

provides an opportunity as well as a challenge to query very large graphs for obtaining insights into the data set as well as extract desired information using an expressive query. Note that search is different from querying and allowing comparison operators and wild cards in a query increases its expressiveness. With the size of the graph databases growing steadily and with the users' need to search for complex patterns, expressing them as graph queries and retrieving all or some answers is very much needed. Hence, approaches for processing queries on large graphs have become a fundamental task (in addition to substructure mining, community and hub detection etc.) for retrieving answers efficiently and effectively with respect to user specified patterns in the form of queries.

User specified patterns (as queries) can vary from completely known patterns (where user exactly knows what s/he is looking for, also termed as an exact match of a subgraph) to include range specification and comparisons, unknown patterns with wild card specification on node and edge labels as well as edge distances in a query. Moreover, the user may also be interested in uncovering results which are combination of multiple patterns (using Boolean operators.) Similar to database querying, this calls for operators (both comparison and logical) to be available for graph querying to satisfy user requirements. Although several techniques for graph querying have been proposed [4, 7, 10, 11, 13, 14, 17] none of them support (expressive) queries.

Most of the extant querying or search approaches on graphs either materialize the graph in main memory or keeps it on disk to be staged into memory as needed. Disk-based approaches are good to deal with graph sizes larger than possible in memory, but introduce difficulties for providing customized buffer management and introduces I/O latency. In contrast, partitioned approaches overcome this by partitioning a graph into desired-sized partitions each of which can fit in memory, but introduces complexity in terms of correctness and efficiency in terms of the number of partitions loaded. This approach also benefits from not using the partitions that are not needed for processing queries. This approach has the potential for figuring out the sequence in which partitions need to be loaded for processing (whether one at a time or k at a time) using the characteristics of the partitions generated and the queries. This approach can also accommodate different processor characteristics by matching partition sizes to processor memory and computation capacities. Finally, this approach is amenable to parallel processing of partitions as well. The numerous benefits of a partitioned approach have motivated the approach presented in this paper.

In the presence of partitions, query answering starts from a particular partition (or partitions) and based on the query plan and may need additional partitions to evaluate a query. The starting node of a query could be present in multiple partitions necessitating a way to rank all the starting partitions and choose one (or top k if multiple partitions are used in parallel.) To make things even more challenging, the answer(s) of a query may span multiple partitions requiring us to *load* different partitions in a proper sequence and the same partition more than once in that sequence. This calls for techniques to analyze the choice of

partitioning strategies and partition characteristics to determine the order of loading partitions.

In the ideal case, we want to load the minimum number of required partitions to answer a query (or a batch of queries.) A required partition is one in which one of the query plan node exists. The lower bound is loading at most the number of required partitions *only once*. Note that the lower bound cannot be determined for a query from analyzing the partitions and the query plan chosen because whether an answer spans multiple partitions can only be determined at run time. Hence, we propose a number of metrics to base our heuristics for choosing partitions after each iteration (an iteration is the processing of 1 or k partitions in parallel) to minimize the total number of partitions loaded. This can also be done for a batch of queries rather than each query separately. Note that the number of partitions loaded may be greater than the number of distinct partitions loaded.

This paper focuses on two aspects of the above problem: (i) correctness of query evaluation for partitioned graphs and (ii) proposing and evaluating heuristics for reducing the number of partitions loaded for evaluating a query to produce all answers. This paper builds upon the existing main memory QP-Subdue [3] graph processing system and extends it to work on graph partitions. QP-Subdue uses a cost model for generating query plans and chooses the minimum cost query plan for execution. As the entire graph is loaded into main memory, all start nodes are expanded independently to obtain all answers. In this paper, we use two widely-used partitioning strategies and identify several metrics for partition evaluation. Furthermore, the paper uses a mix of query and partition characteristics to determine initial partition for loading and after each partition execution. The contributions of the paper are:

- First attempt at processing queries using graph partitions for scalability
- Extend a main memory graph query processing algorithm and establish its correctness (Sects. 3, 4 and 6)
- Metrics and heuristics using graph partition information and query characteristics to reduce the number of partitions loaded for query evaluation (Sect. 5)
- Extensive experimental analysis that validates proposed heuristics on real-world and synthetic data sets using a broad range of queries (Sect. 7).

Remainder of the paper is organized as follows. Section 2 summarizes relevant work. Section 3 summarizes query/graph representation, cost-based query plans, and the partitioning approaches used. In Sect. 4, we describe the PGQP (Partition-based Graph Query Processor) architecture and correctness. Section 5 discusses the metrics computed and proposed heuristics. Section 6, briefly describes the implementation of PGQP. Section 7 demonstrates the viability of this approach along with the experimental validation of proposed heuristics. Conclusions are in Sect. 8.

2 Related Work

Searching/Querying: Querying/searching is useful for retrieving information for understanding the contents of graph databases. The process of finding exact/similar patterns in graphs is a well researched area. In Graph-grep [5], a variable path index based approach is used. A separate hash index is constructed for different path lengths containing all possible paths up to length l starting from each node in the graph database. In G-index [16], frequent substructures are indexed as a prefix tree by translating the graph into unique edge sequences (called canonical labels) using depth first search (DFS) coding. Using such an index, a search will produce results only if it is frequent. Another technique for query processing, G-Ray [15], expands a seed node by finding a matching node followed by bridging both nodes by the best possible path, while proposing a goodness score quantifying the proximity between two nodes that is used to rank the results. The main challenges of the graph query/search techniques discussed above are managing the size of the index in Graph-grep or the size of canonical labels as a prefix tree (G-Index) or information about remaining vertices (G-Ray). The response time increases with respect to the size of the graph. Also, none of the above support a query in its generality.

Recently, there have been some attempts to go beyond search. QP-Subdue [3] is an attempt to move towards general purpose querying of graphs. It is a main memory approach and has modified a substructure mining algorithm to perform query evaluation. A cost-based plan generator is used. Relational and Boolean operators as well as some wild cards are supported for query specification. Querying by example [7] is another attempt to work only with RDF tuples. Moreover, it does not handle wild cards and general graph query and is also limited by the size of main memory.

Partitioning Schemes: Graph partitioning, addressed extensively is to partition a graph into p (roughly equal) partitions. As there are many ways to partition a graph, some metric such as minimizing the number of edges between the partitions (termed cut set) is commonly used. Many large scale graph partitioning tools are available. For this work, we use two widely-used systems – METIS [8] and Karlsruhe High Quality Partitioning (**KaHIP**) [12] – for partitioning a graph on which queries are processed. METIS uses a multilevel algorithm proposed by Chaco [6]. It provides high quality partitions and supports four different heuristics during the coarsening phase to accommodate different types of graphs. KaHIP is a family of graph partitioning programs and uses various local and global search techniques, different coarsening strategies as well as several meta-heuristics. KaHIP provides very high quality partitions and makes a good trade-off between quality and run time. KaHIP implements novel local improvement schemes to fit most kind of graphs such as continental-sized road networks as well as large social networks and web graphs compared to METIS. Although graph partitioning have been used for mining [2], using them for graph querying is still an ongoing process. This paper uses available partitioning schemes for evaluation and extends the main memory query processing approach to a partitioned approach and analyzes several heuristics.

3 Graphs, Queries, Plan Generation, and Partitioning

In this paper, we divide a graph into desirable-sized partitions and process queries by loading the partitions one at a time as needed to obtain all or k answers. For partitioning a graph, we use two popular approaches – METIS and KaHIP (each comes with their own heuristics for partitioning.)

Table 1. Graph representation for partition P_2

(a) Vertex Table

vID	vL	pID
5	Beyond All Boundaries	1
6	2011	2
7	Drama	2
8	Year	2
9	Movie	2
10	Genre	2

(b) Edge Table

dir	s_vID	d_vID	eL
u	5	6	In year
u	5	7	Genre is
u	5	9	is
u	6	8	is
u	7	10	is

Graph Representation: Both undirected and directed graphs can be used for our approach. We follow the Subdue [9] representation where vertices are input as an unordered sequence of <vertex id (vID), vertex label (VL)> pairs and edges are input as unordered tuples of <direction (dir as d: directed, u: undirected), source vertex id (s_vID), destination vertex id (d_vID), edge label (eL)>. For this approach, a partition number (pID) is added to each vertex in the above representation to identify query answers crossing partition boundaries.

Figure 1a shows an example of an IMDB (movie database) graph with type node information like “Person”, “Genre”, etc. The two graph partitions P_1 and P_2 for this graph along with the replicated cut set edges are shown in Figs. 1b and c, respectively. The broken lines show the additional information kept as

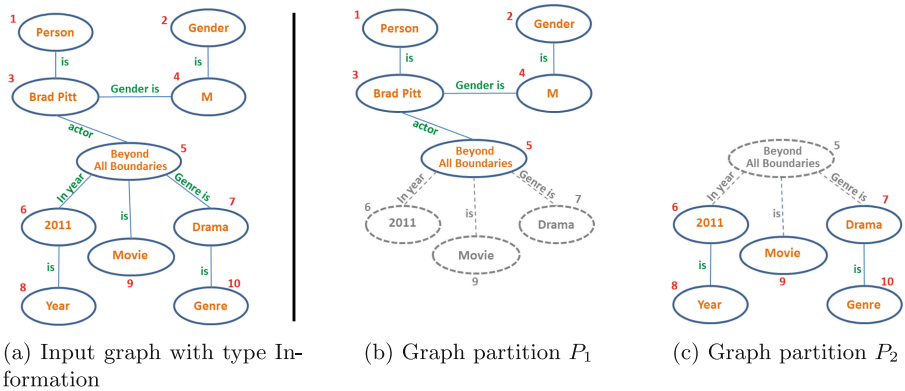


Fig. 1. Sample input graph and its partitions

part of each partition that corresponds to the cut set. This extension of each partition is necessary for keeping track of answers that span multiple partitions. Tables 1a and b show the vertex and edge table representation, respectively, for the partitioned IMDB graph shown in Fig. 1c. This is used as the graph on which a query is processed. Figure 2 illustrates the representation of a sample query - “Find all actors in the movie ‘Beyond all boundaries’ and year of its production”.

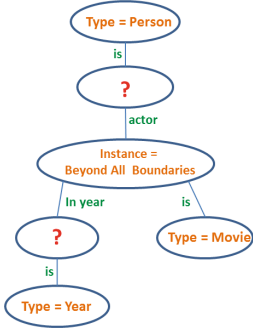


Fig. 2. Representation of a query on the input graph shown in Fig. 1a

Cost-Based Plan Generation: Along the lines of relational approach to query processing where meta-data collected from the database is used to estimate the cost of a query plan, we create a graph catalog which contains information that are relevant to plan generation in a graph database. The graph catalog (generated by making a single pass over the graph database) consists of information, such as type cardinality, average instance cardinality, average connection cardinality, min and max values of type nodes. For more details on this refer to [3]. Cost formulas have been derived for conditions and graph traversals using the information in the catalog. A query plan generator generates the search space of plans (each node in a query being a potential start node for query evaluation) along with the number of estimated substructures generated as the cost. A branch and bound algorithm is used to identify a “good” plan for a given graph query. It could be multiple plans whose results are combined to obtain all answers.

the number of estimated substructures generated as the cost. A branch and bound algorithm is used to identify a “good” plan for a given graph query. It could be multiple plans whose results are combined to obtain all answers.

Graph Partitioning: We use METIS and KaHIP for partitioning and use many of the configurations supported to understand their effect from a query evaluation perspective. Currently, there are no partitioning schemes that generate varying partition sizes. In this paper, we have used 2 configurations of METIS - (i) kway as partitioning type and sorted heavy edge matching as coarsening type (*kway_shem*) and (ii) recursive bisection as partitioning type and sorted heavy edge matching as coarsening type (*rb_shem.*) We have used 4 configurations of KaHIP to partition the graph - *fast*, *eco*, *fastsocial* and *ecosocial*. These 6 strategies are used to partition an input graph into 4 partitions.

4 Partitioned Approach to Query Processing

Processing queries on a partitioned graph is very different as compared to processing queries on a single graph. When a graph is partitioned it will generate k graphs (G_0, G_1, \dots, G_k) such that all the k partitioned graphs can be combined to form the original graph. Some additional information is needed (e.g., cut set) for combining graphs. If all the answers of a query are in a single partition, it is no different from a non-partitioned approach. However, the case where answers span multiple partitions need to be computed correctly. This entails adding some additional information to each partition to keep track of partial answers and

continuing them in appropriate partitions. These continuations may also require visiting the same partition more than once. This again needs to be handled properly to ensure correctness of results.

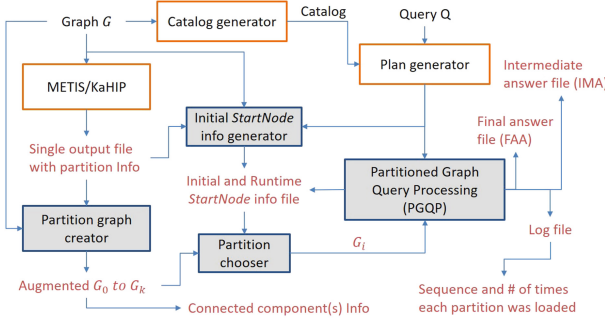


Fig. 3. The architecture of the PGQP system

is meaningful if the number of partitions are large and hence can benefit from parallelism. Choosing k partitions to process (instead of 1) is more challenging in terms of book keeping (and/or interprocess communication) and is also prone to reloading the same partition more than once. This is beyond the scope of this paper and hence is not discussed further.

4.1 PGQP System Architecture

Input to the PGQP system is still a graph database and one or more queries. QP-Subdue architecture has been extended (shown in Fig. 3) to accept a partition with additional cut set information (instead of a complete graph) and process a query from a set of specified starting points. Our approach uses files for communication between partition executions. At the end of the processing of each partition appropriate information is written for continuing query processing using other partitions. The shaded modules in Fig. 3 are extensions for the partitioned approach. The non-shaded modules correspond to pre-existing systems/modules that are used, such as METIS/KaHIP, catalog generator, and plan generator. The partition chooser module chooses the next partition to processing using a specified heuristics or base line.

As METIS and KaHIP generate partitions following their own representations, the partition graph creator is implemented to convert the output into desired input for PGQP module as well as compute metrics in one pass. Three metrics are computed by the system: (i) number of connected components in each partition (one time), (ii) number of start nodes in each partition (one time) and its update after processing a partition, and (iii) sequence and number of times each partition is loaded (at the end) for query evaluation. These are used in various ways as described in the next sections.

In this paper, we process one partition at a time. Our correctness is based on minimal extensions to each partition and keeping track on intermediate results after each partition processing and continuing them as needed. However, it is also possible to process multiple partitions in parallel. This approach

4.2 Correctness of the Approach

The three cases that need to be addressed for answering queries correctly are shown in Fig. 4. We establish correctness for each case and show how they are implemented in Sect. 6. When query results are completely inside a single partition as in Fig. 4a, while processing that partition, all results in that partition are computed and stored.

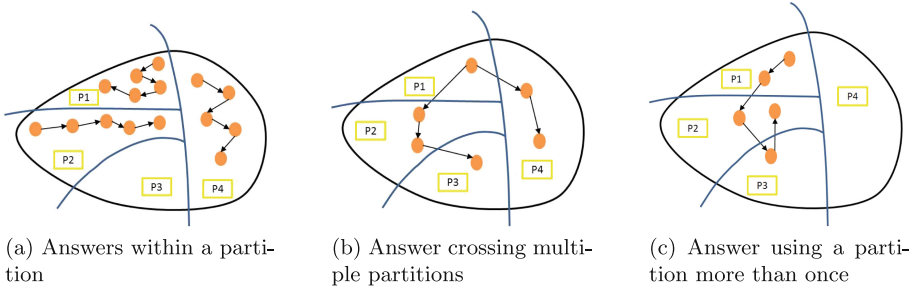


Fig. 4. Different cases of query evaluation in partitioned graph

This is equivalent to QP-Subdue running individually on a partition as complete graph. For the case where query results span multiple partitions (illustrated in Fig. 4b), when a query expands into another partition, the intermediate answers are written into the a partition’s continuing answers file. Importantly, we update the starting node information in the new partitions that the answers span into using the one edge cut set information that has been added to each partition (see Figs. 1b and c). All files are written at the end of processing a partition. When the next partition is loaded (by the partition chooser module based on the heuristic), the SNI file is used to identify all starting nodes (both start nodes and continuing nodes) for processing. Things that ensure correctness are: (i) addition of cut set information to each partition to correctly identify the start nodes in each *continuing partition*, (ii) update of SNI file and its usage to determine all start nodes (whether initial or continuation) in that partition, and (iii) Carrying and concatenating intermediate results as new partitions are processed. Figure 4c is a special case of the above when an answer instance comes back to a partition that has already been processed.

Since a query result can span multiple partitions or even need the same partition more than once, sequencing of partition loads is important from a performance standpoint which is the subject of Sect. 5.

5 Metrics and Heuristics for Partitioned Query Evaluation

One of the widely used metrics for graph partitioning is the *cut set*. Partitioning strategies (e.g., METIS and KaHIP) and partitioning schemes (e.g., kway_shem

of METIS, eco of KaHIP) try to minimize the cut set to reduce inter-partition connections. *In our approach, cut sets are the means by which an answer to a query can span multiple partitions.* In general, if the number of edges in a cut set-cut is small, the likelihood of an answer crossing to another partition from that partition is likely to be small as well. Of course, this depends on the query mix and the characteristics of the *cut set* (e.g., connecting nodes and labels.) Since our focus is on computing query answers across partitions correctly by loading partitions one at a time, we want to identify metrics and formulate heuristics to reduce the number of partition loads, the ideal being the minimum required.

5.1 Number of Query Plan Start/Continuation Nodes in a Partition

For a partitioned approach to query evaluation, we first identify the partitions in which a *query plan start node* exists (termed starting partitions.) Since there can be many start nodes in each starting partition, it is useful to compute the number of such nodes in each starting partition. This is done by using the label information of the start node. It is also possible to rank partitions based on the number of start nodes. If a partition does not have a start node, it may be needed only if an answer spans that partition. In this paper, we propose two heuristics for choosing a partition to process based on the number of query plan start/continuation nodes in each partition. Note that any query plan node can become a continuing node in a partition during the process of query evaluation. Starting partition for processing a query is determined based on the chosen heuristic. Since these heuristics involve number of query plan start nodes in a partition, *this could change after processing a partition.* Hence, this metric is updated at the end of each partition processing to determine the next partition to be used. This process is continued until there are no more partitions to process. The two heuristics, based on the number of start nodes (SN), are discussed below.

1. **MAX-SN-heuristic:** This is similar to the greedy strategy for the choice of the next partition for processing. The partition with the most number of start nodes (from the eligible set of partitions) is chosen. If there is a tie, one is chosen randomly. The intuition behind this heuristic is that highest number of query answers (due to maximum number of start nodes) will be explored and will help if they span partitions. Also, if some answers are answered within that partition (initially or while continuing), it will reduce the number of spans. The start node information is updated at the end of each partition processing and the process repeated. Note that a previously processed partition may have to be processed again.
2. **MIN-SN-heuristic:** In this case, we load the partition with the least number of start nodes in that partition. Again, ties are resolved randomly. The intuition behind this heuristic is that we accumulate the spanning requirements into partitions with larger number of start nodes in the hope that they can be processed only once.

In order to evaluate the proposed heuristics, we need a baseline. We use random way of choosing the initial as well as the next partition from among the

set of eligible partitions as our baseline. We completely ignore the number of start nodes in any partition. We believe that the MAX-SN-heuristic will lead to better performance (in terms of the number of partition loads) than the other heuristic and the baseline. We also believe that either heuristic should do better than the baseline choice (**RANDOM-SN.**) When the number of partitions are large, the impact of the proposed heuristics is likely to be more pronounced than when compared to the baseline. These observations have been evaluated for validation in Sect. 7.1.

5.2 Total Number of Connected Components

Partitioning strategies seem to focus more on the *cut set* and not worry so much about the *number of connected components* generated in each partition. However, for query processing, the number of connected components within each partition and hence the total number of connected components in a partitioning scheme are very important. The partitioning schemes typically produce one partition with a single connected component and the rest of the partitions with varying number of connected components in each of them. We compute the number of connected components in each partition for each partitioning scheme during the same pass in which we compute other metrics.

In the presence of disconnected components in a partition, if a query answer spans more than one connected component in that partition, it forces this partition to be *loaded again* for processing. This is avoided if a partition has only one connected component and this possibility is reduced if the number of connected components is small within a partition.

Instead of analyzing the effect of connected components at the individual partition level, we evaluate the effect of total number of connected components in a partitioning scheme on the performance (i.e., the number of partitions used) of query processing. We believe that *choosing a partitioning scheme with least number of total connected components* (termed **MIN-CC-heuristic**) is always better from a query evaluation perspective as compared to any other scheme and especially one that produces highest number of total connected components (termed **MAX-CC-heuristic**). This is likely to make an answer stay inside a partition more often and thereby reduce the total number of partitions used. We will also evaluate this with experiments showing total partition loads using different partitioning schemes of the same graph in Sect. 7.2. Moreover, either of these heuristics will perform better than arbitrarily choosing any of the partitioning schemes randomly as baseline (**RANDOM-CC.**)

5.3 Quantitative Measures for Evaluating the Heuristics

In our approach, the total time to answer a query will be directly proportional to the number of partitions loaded (*includes multiple loads of the same partition*) in order to generate all answers. For each query, the ideal case can be inferred from the number of partitions in which a query plan node occurs (actually an upper bound.) We use this as the lower bound on the number of partitions

that are needed for processing a query. A quantitative measure for evaluating a heuristic h (e.g., MIN-CC) can be derived by comparing the lower bound (L_{ideal}) of partitions to the number of actually loaded partitions (AL_h). This load ratio of ideal to actual (L_{ideal}/AL_{MAX}) will indicate the effectiveness of a heuristic. This value is at best 1. A heuristic that has a higher value for this ratio is better than the one with a lower value.

For this purpose, we define two measures for MIN-SN and MAX-SN heuristics (denoted by h below) – one that measures the average load ratio for the *same query across partitioning schemes* for a given database D ($\mathbf{h}(D)_{pschemes}^{query}$) and another that measures the average load ratio for a *batch of queries using the same partitioning scheme* for a given database D ($\mathbf{h}(D)_{qbatch}^{pscheme}$). For example, $\mathbf{MAX-SN(IMDB)}_{pschemes}^{Q1}$ can be expressed as $\frac{1}{|pschemes|} \sum_{pschemes} (L_{ideal}/AL_{MAX})$, where AL_{max} is the number of actual partitions loaded for Q1 using MAX-SN-heuristic. Others are defined similarly.

The second measure defined above is also used for the connected component heuristics: **MIN-CC-heuristic** and **MAX-CC-heuristic**. For any of the starting node related heuristics, MIN-CC-heuristic is the average load ratio of a batch of queries executed on the partition scheme with minimum number of connected components. MAX-CC-heuristic is defined similarly. These measures are computed for our experiments to validate our conjecture.

Note that CC and SN heuristics are orthogonal or independent of each other. Either can be used individually to improve performance and they can be used in tandem as well for improving the performance further. This provides alternatives to the user depending upon the graph characteristics and the partitioning strategies available.

6 Implementation Summary

Only the important aspects of implementing the PQGP system (whose architecture is shown in Fig. 3) is briefly summarized in this section. For details, refer to [1]. We capture start node and its label information of the query plan for each partition in a **Starting Node Information (SNI)** file. This file is updated at the end of processing each partition and used for choosing the next partition based on a heuristic as well as identifying the start nodes in that partition. In addition, we keep an **Intermediate Answers (IMA)** file – *one for each partition*. This file stores intermediate or partial query results relevant to that partition. When a query has been answered completely the final results are appended to a **Final All Answers (FAA)** file. This file contains all the answers at the end of query processing. Finally, we also keep a **log file** to compute some run time metrics needed for our analysis.

6.1 Management of Partial Results

For each query, based on the start node label of the query plan, PGQP finds all the relevant partitions containing the start node label, number of occurrences

(or nodes) for that label and generates an initial SNI file. The initial SNI file contains the starting vertex label (according to the query plan) and the number of its occurrences. Figure 5 shows the initial SNI for all partitions with starting node label and number of occurrences. Note that the vertex ids are NULL indicating that they are *start nodes and not continuation nodes*. Continuation nodes will have both a label and a vertex id obtained from the partition extension information. Following the chosen heuristic (let us assume MAX-SN for this discussion), we pick partition 1 (P1) as our starting partition to process. Expansion process is identical to breadth first search from the starting vertex id and abiding by the query plan.

Assume that some answers can be found completely in P1 and some continue in other partitions. At the end of processing P1, complete answers are written into the FAA file (here 1-17-50-201). Final answers can easily be demarcated from partial answers by the size of the answer and edge label(s). Following the query plan, a result having the same size and edge label(s) of the query is a complete answer while any answers with a lesser size is considered a partial result. Partial answers continue in other partitions which are written into corresponding IMA files. For example if 1-4 crossed P1 and moved into P2 this intermediate result is written in *IMA₂* while the SNI has been updated with the vertex id 4 for B. The SNI after completion of P1 indicates that, query answers can continue from partial answers in P2 and P3 (the ones with vertex ids) or start from node labels A in partitions 2 and 3 (marked with NULL in vertex ids). The vertices already expanded are dropped from the SNI. Continuing with our MAX-SN heuristic we load P2 next as it has 18 vertices in SNI as compared to 10 of P3.

Query answers that start in P1 and end in P2 are written to the FAA. See that some query answers continued onto other partitions and the SNI file was updated accordingly. The intermediate results going now into P3 are appended to *IMA₃*. See in Fig. 5, *IMA₃* is updated by containing both substructures with 2 and 3 vertices. All of these intermediate results will be expanded when P3 is loaded. When all answers are computed, the SNI file will be empty indicating that there are no more partitions to be processed.

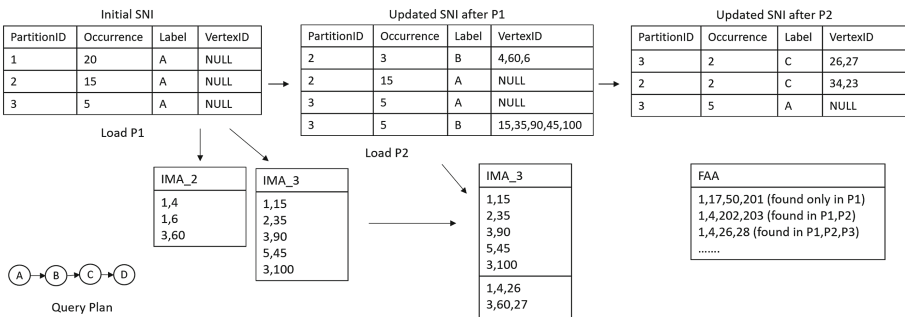


Fig. 5. Query answering in the PGQP system

The SNI file and all IMA files are dropped at the end of query processing. The FAA file contains all the answers added incrementally as soon as an answer is completed. Hence, the user can see results as and when they are generated instead of waiting for the processing to complete. This process can be stopped to generate only k answers.

A combination of the SNI, IMA and FAA guarantees correctness. Intuitively SNI contains all starting points or continuation points for partial results while the IMA contains the intermediate or partial results for each partition. After every iteration, the partitions that needs processing have a non-empty IMA file. The SNI file is properly updated to facilitate easy continuation of the query from the next partition.

7 Experimental Analysis

Experimental results presented in this section provide an analysis of the proposed heuristics and their validation.

Experimental Setup: All experiments have been carried out on Dual Core AMD Opteron 2 GHz processor machine with 16 GB memory. To test the *correctness of our approach*, query results given by QP-Subdue [3] - a non-partitioned, main-memory query processor for graph databases - have been used as ground truth. The largest graph size we have been able to handle in QP-Subdue on our 16GB machine is 550 K nodes and 1700 K edges.

Datasets: We have used two datasets: (i) The Internet Movie Database - **IMDB** graph (1750 KV, 5100 KE) containing information of movies, actors, genres, year, company, etc. The vertex labels in IMDB are unique, thus the result set for any query is small (some times one.) (ii) A **synthetic graph** generated using Subgen with 400 K vertices, 1200 K edges, 2000 unique vertex labels and 4000 unique edge labels with uniform distribution to analyze queries with multiple results.

Query Characteristics: Three different queries were used on IMDB - Find tv-series and their production companies from the animation **AND** comedy genres that involved “Kelsey Wagner” (*Query 1*), List the “Adam Sandler” movies and their production companies that belonged to the comedy **AND** Sci-Fi genres but the release year was **NOT EQUAL** to 2000 (*Query 2*) and, Find all the production companies where “Fred Wolf” has worked as a writer **OR** “Salma Hayek” has worked as an actress (*Query 3*). These formulated queries have different characteristics that are relevant to the partitioning problem, such as query answers completely inside a single partition (*Query 3*), query answers spanning multiple partitions for exact results (*Query 2*), and queries that need to use the same partition more than once (*Query 1*). Moreover, we have used different combinations of comparison operators ($<$, \leq , $>$, \geq , \neq , $=$) and logical operators (OR, AND). Query plans were generated for all the queries on IMDB data sets using QP-Subdue.

We have used the synthetic graph mainly to test our metrics and heuristics for multiple answers spanning several partitions. Due to the synthetic nature of the

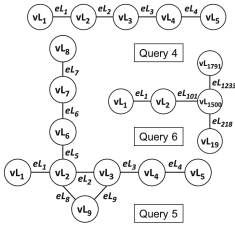


Fig. 6. Queries for synthetic graph

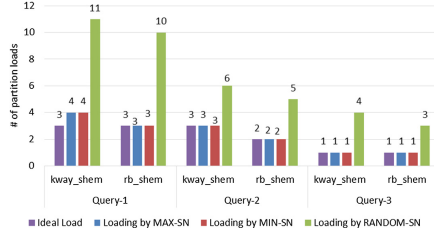


Fig. 7. Performance of query answering on the IMDB graph partitioned by METIS

graph, embedded substructures were used instead of queries that not only span partitions but also need a partition more than once. In the synthetic graph, we embed 200 instances of a substructure shown in Fig. 6 that also acted as Query 5. One of the queries was a subgraph of the embedded structure (Query 4), whereas for one query *only* a part (2 nodes and one edge) of the embedded substructure was present in the synthetic graph (Query 6.) Each query was executed using the MIN-SN, MAX-SN and baseline RANDOM-SN choice, on a dataset that was partitioned by 6 schemes, resulting in a total of 54 experiments per dataset.

7.1 Evaluation of Start Node Heuristics

Figures 7 and 10 illustrate the number of partitions loaded for the ideal case, proposed heuristics (MAX-SN and MIN-SN) and the baseline (RANDOM-SN) for IMDB partitioned by METIS or KaHIP, respectively. From these figures, we obtain the values of the evaluation measures - $h(\mathbf{D})_{pschemes}^{query}$ and $h(\mathbf{D})_{qbatch}^{pscheme}$ (defined in Sect. 5.3) listed in Tables 2 and 3, respectively. The higher values for the proposed MIN-SN or the MAX-SN heuristic show that they have a better performance while answering single or a batch of queries on a partitioned graph as compared to the baseline RANDOM-SN choice. Similar inference has been made for the Synthetic graph, for which the values of the measures have been calculated from Figs. 8 and 9.

Table 2. Performance of SN heuristics for a single query across partitioning schemes

$h(\mathbf{D})_{pschemes}^{Query_i}$	IMDB		
	i = 1	i = 2	i = 3
MAX-SN	0.847	0.944	1.0
MIN-SN	0.847	0.944	1.0
RANDOM-SN	0.302	0.337	0.320
	Synthetic		
	i = 4	i = 5	i = 6
MAX-SN	0.587	0.545	0.618
MIN-SN	0.385	0.368	0.401
RANDOM-SN	0.280	0.260	0.270

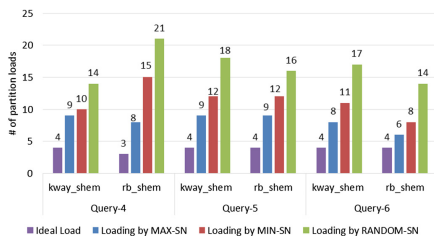
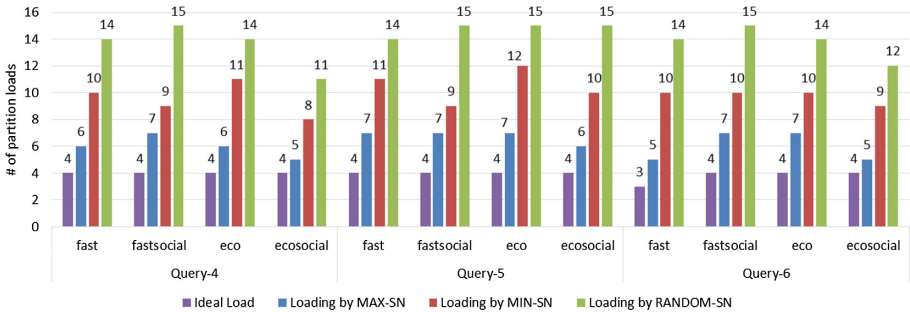


Fig. 8. Performance of query answering on the synthetic graph partitioned by METIS

Table 3. Performance of SN heuristics for a batch of queries on partitioning schemes

$h(D)_{\{Q1, Q2, Q3\}}^{pscheme_i}$	IMDB					
	fast	fastsocial	eco	ecosocial	kway_shem	rb_shem
MAX-SN	0.889	0.889	0.889	1.0	0.917	1.0
MIN-SN	0.889	0.889	0.889	1.0	0.917	1.0
RANDOM-SN	0.262	0.317	0.345	0.306	0.341	0.344
$h(D)_{\{Q4, Q5, Q6\}}^{pscheme_i}$	Synthetic					
	fast	fastsocial	eco	ecosocial	kway_shem	rb_shem
MAX-SN	0.613	0.571	0.603	0.756	0.463	0.495
MIN-SN	0.355	0.430	0.366	0.448	0.366	0.344
RANDOM-SN	0.262	0.267	0.279	0.321	0.248	0.226

**Fig. 9.** Performance of query answering on the Synthetic graph partitioned by KaHIP

Further, Tables 2 and 3 also show that the *average load ratio for MAX-SN is greater than or equal to the average load ratio of MIN-SN heuristic, for processing a single or batch of queries across all or a single scheme, respectively.* Therefore, these experiments also validate our claim that the **MAX-SN heuristic performs as good as or better than the MIN-SN heuristic.** In case of Synthetic graph, where multiple results for each query are possible MAX-SN is always better than MIN-SN (Figs. 8 and 9). However, due to presence of only unique vertex labels the order and number of partition loads for IMDB queries is same for both MIN-SN and MAX-SN (Figs. 7 and 10).

7.2 Evaluation of Connected Components Heuristics

The heuristic for choosing a partitioning scheme based on connected components metric is for improving performance.

Table 4 shows the evaluation of the partitioning schemes that generated the highest and least number of total connected components by measuring their performance for a batch of queries ($h(D)_{qbatch}^{pscheme}$). In most of the scenarios,

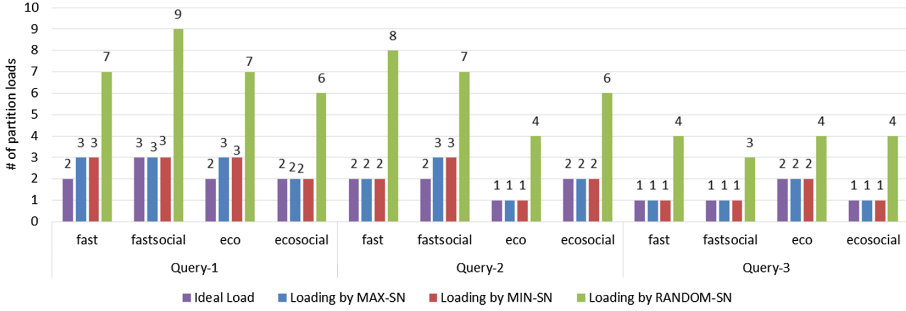


Fig. 10. Performance of query answering on IMDB graph partitioned by KaHIP

the higher values for the schemes that generate the least number of total connected components, irrespective of the MAX-SN or MIN-SN heuristic, validate our claim that choosing a scheme using MIN-CC heuristic serves as better choice for answering queries on partitioned graphs as compared to MAX-CC heuristic.

Table 4. Performance of CC heuristics on a batch of queries

$h(D)_{\{Q1, Q2, Q3\}}^{pscheme_i}$	IMDB			
	KaHIP		METIS	
	MIN-CC	MAX-CC	MIN-CC	MAX-CC
	ecosocial	fast	rb_shem	kway_shem
MAX-SN	1.0	0.889	1.0	0.917
MIN-SN	1.0	0.889	1.0	0.917
# Total CC	40975	77687	58371	80417
$h(D)_{\{Q4, Q5, Q6\}}^{pscheme_i}$	Synthetic			
	KaHIP		METIS	
	MIN-CC	MAX-CC	MIN-CC	MAX-CC
	ecosocial	fast	kway_shem	rb_shem
MAX-SN	0.756	0.613	0.463	0.495
MIN-SN	0.448	0.355	0.366	0.344
#Total CC	5367	14365	16606	17316

However, if the difference between highest and the least number of total connected components is not significant then no definite choice can be made. This case is exemplified by the MET-IS partitioned Synthetic graph in Table 4 where both the listed schemes have similar performance because the difference between the number of total connected components produced by them falls below 5%. Therefore, the ecosocial partitioning scheme that always generates substantially low number of total connected components as compared to the other three schemes, becomes the preferred partitioning scheme using MIN-CC heuristic.

Thus, Sects. 7.1 and 7.2, empirically validate that MAX-SN and MIN-CC independently are better than other heuristics. Furthermore, they can be combined to improve the performance even more as can be seen from Table 4.

8 Conclusions

In this paper, we have proposed an approach for processing queries over partitions of a graph database. This is one way to ensure scalability of query processing to a graph database of any size. We have addressed correctness and proposed

heuristics to reduce the amount of work done (in terms of the number of partitions loaded) for query processing. Implementation of the proposed approach as well as validation of the proposed heuristics has been shown on real-world and synthetic data sets.

Beyond this, use of parallel processing of partitions, heuristics for a batch of queries, and partitioning strategies to reduce the number of connected components are being explored. Use of Map/reduce approach is another possibility as well.

References

1. Bodra, J.: Processing queries over partitioned graph databases: an approach and it's evaluation. Master's thesis, The University of Texas at Arlington, May 2016
2. Das, S., Chakravarthy, S.: Partition and conquer: map/reduce way of substructure discovery. In: Madria, S., Hara, T. (eds.) DaWaK 2015. LNCS, vol. 9263, pp. 365–378. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22729-0_28
3. Das, S., Goyal, A., Chakravarthy, S.: Plan before you execute: a cost-based query optimizer for attributed graph databases. In: Madria, S., Hara, T. (eds.) DaWaK 2016. LNCS, vol. 9829, pp. 314–328. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-43946-4_21
4. Fan, W., Li, J., Ma, S., Wang, H., Yinghui, W.: Graph homomorphism revisited for graph matching. PVLDB **3**(1), 1161–1172 (2010)
5. Giugno, R., Shasha, D.: Graphgrep: a fast and universal method for querying graphs. In: ICPR, vol. 2, pp. 112–115. IEEE Computer Society (2002)
6. Hendrickson, B., Leland, R.: A multilevel algorithm for partitioning graphs. In: Proceedings of the 1995 ACM/IEEE Conference on Supercomputing, Supercomputing 1995. ACM, New York (1995)
7. Jayaram, N., Khan, A., Li, C., Yan, X., Elmasri, R.: Querying knowledge graphs by example entity tuples. IEEE Trans. Knowl. Data Eng. **27**, 2797–2811 (2015)
8. Karypis, G., Kumar, V.: Multilevel graph partitioning schemes. In: Proceedings of the 1995 International Conference on Parallel Processing. Algorithms & Applications, vol. III, Urbana-Champaign, Illinois, USA, 14–18 August 1995, pp. 113–122 (1995)
9. Ketkar, N.S., Holder, L.B., Cook, D.J.: Subdue: compression-based frequent pattern discovery in graph data. In: Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations, OSDM 2005, pp. 71–76. ACM, New York (2005)
10. Khan, A., Li, N., Yan, X., Guan, Z., Chakraborty, S., Tao, S.: Neighborhood based fast graph search in large networks. In: SIGMOD Conference, pp. 901–912 (2011)
11. Mongiovi, M., Di Natale, R., Giugno, R., Pulvirenti, A., Ferro, A., Sharan, R.: Sigma: a set-cover-based inexact graph matching algorithm. J. Bioinform. Comput. Biol. **8**(2), 199–218 (2010)
12. Sanders, P., Schulz, C.: Engineering multilevel graph partitioning algorithms. In: Demetrescu, C., Halldórsson, M.M. (eds.) ESA 2011. LNCS, vol. 6942, pp. 469–480. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23719-5_40
13. Tian, Y., Patel, J.M.: Tale: a tool for approximate large graph matching. In: ICDE, pp. 963–972 (2008)
14. Tong, H., Faloutsos, C., Gallagher, B., Eliassi-Rad, T.: Fast best-effort pattern matching in large attributed graphs. In: KDD, pp. 737–746 (2007)

15. Tong, H., Faloutsos, C., Gallagher, B., Eliassi-Rad, T.: Fast best-effort pattern matching in large attributed graphs. In: Berkhin, P., Caruana, R., Wu, X. (eds.) KDD, pp. 737–746. ACM (2007)
16. Yan, X., Yu, P.S., Han, J.: Graph indexing: a frequent structure-based approach. In: SIGMOD Conference, pp. 335–346 (2004)
17. Zhang, S., Yang, J., Jin, W.: Sapper: subgraph indexing and approximate matching in large graphs. PVLDB **3**(1), 1185–1194 (2010)



Querying Heterogeneous Data in Graph-Oriented NoSQL Systems

Mohammed El Malki^{2(✉)}, Hamdi Ben Hamadou¹, Max Chevalier¹,
André Péninou², and Olivier Teste^{2(✉)}

¹ Université Toulouse 3 Paul Sabatier, IRIT (CNRS/UMR5505),
Toulouse, France

² Université Toulouse 2 Jean Jaurès, UT2C, IRIT (CNRS/UMR5505),
Toulouse, France

{elmalki, prenom.nom, teste}@irit.fr

Abstract. NoSQL systems are based on a “*schemaless*” approach that not does require schema specification before writing data, allowing a wide variety of representations. This flexibility leads to a large volume of heterogeneous data, which makes their querying more complex for users who are compelled to know the different forms (i.e. the different schemas) of these data. This paper addresses this issue focusing on simplifying the heterogeneous data querying. Our work specially concerns graph-oriented NoSQL systems.

Keywords: Information systems · NoSQL data stores
Graph-oriented databases · Heterogeneous data querying

1 Introduction

The growing usage of “Not-only-SQL” storage systems, referred as NoSQL, has given ability to efficiently handle large volume of data [4]. Graph-oriented systems are among the most increasingly used NoSQL approaches. A special attention has focused on how to model data in the form of graphs [4]. In this approach, data is represented as nodes, edges and attributes, which allows the modelling of different interactions between data. Graphs modeling is ubiquitous in most social networks, semantic web and bioscience (protein interactions ...) applications.

Graph-oriented systems belong within the “*schemaless*” framework [2, 7], that consists in writing data without any prior schema restrictions; i.e., each node and each edge have its own set of attributes, thus allowing a wide variety of representations [6]. This flexibility generates heterogeneous data, and makes their interrogation more complex for users, who are compelled to know the different schemas of the manipulated data. This paper addresses this issue and consider a straightforward approach for the interrogation of heterogeneous data into NoSQL graph-oriented systems. The proposed approach aims at simplifying the querying of heterogeneous data by limiting the negative impact of their heterogeneity and leads to make this heterogeneity “transparent” for users.

This paper is organized as follows. We highlight in Sect. 2 the issues addressed in this paper. Section 3 gives an overview of the related work. We present in Sect. 4 our

approach for heterogeneous data interrogation in order to simplify data querying. The results of the experimental evaluation of our approach are presented in Sect. 5.

2 Problem Modeling

2.1 Notations

The data modeling in NoSQL graph-oriented systems consists in representing the database as a graph. Figure 1 illustrates an example of a simple graph $G = (V, E, \gamma)$ where $V = \{u_1, \dots, u_n\}$ denotes the set of nodes, $E = \{e_1, \dots, e_m\}$ is the set of edges connecting one node to another and $\gamma : E \rightarrow V \times V$ represents a function that determines the nodes pairs connected by the edges.

The different nodes are described with textual format as presented below:

We can notice in Fig. 1 that the name of the edges can vary (either `To_Write` or `To_Compose`). Similarly, the graph's nodes and their attributes can be heterogeneous.

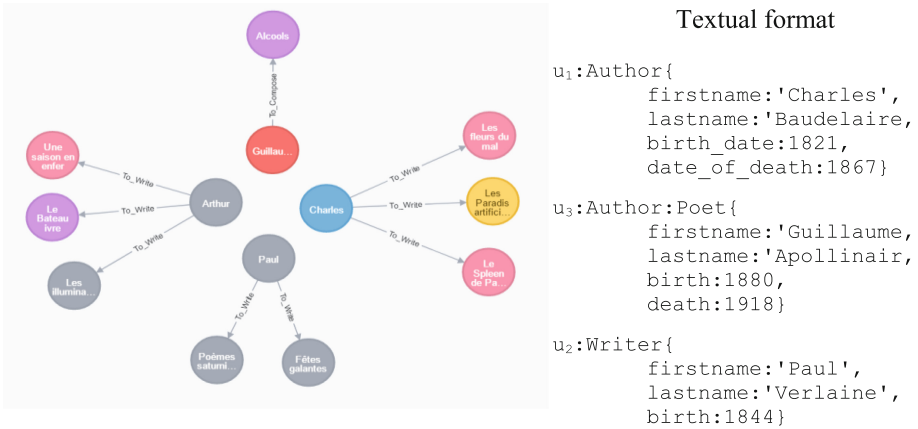


Fig. 1. A graph example

2.2 Heterogeneity Classes

The heterogeneity can be considered from different perspectives [10] depending on the structural elements composing the graph:

Structural heterogeneity refers to data represented by variable structural elements.

Syntactic heterogeneity refers to the fact that a structural element can be referred to a variable way; e.g., the attributes 'birth_date' and 'birth' belong to different nodes but both refer to an author's date of birth.

Semantic heterogeneity defines the problem of two different elements corresponding to the same data, or inversely when a single element is related to two different data; e.g., the edges 'To_Write' and 'To_Compose' both have the same meaning. In this article, we address the different heterogeneity classes discussed in this section.

2.3 Querying Heterogeneous NoSQL Graph

We use Cypher language [4] offer by Neo4j NoSQL graph database to illustrate the problem of querying heterogeneous graphs.

Our model is based on a theoretical foundations, called algebraic core, which ensures the genericity of the approach. In this paper, we only examine the operators defined for projection and selection. This set of elementary operators compose the algebraic core.

Therefore, using naively Cypher language in the context of heterogeneous graphs may leads to produce wrong analyses and to take decisions on incomplete data. In this paper, we propose an approach that allows the user to express a query using the set of attributes, without considering the structural, syntactic, and semantic differences, without changing the original structures of the graphs. Our solution grants the possibility to obtain a “complete” results, without having to explicitly manage the various heterogeneity aspects.

Table 1. Querying heterogeneous data problem

	Projection. <i>Get the lastname, the firstname and the book’s title of the authors</i>	Selection. <i>Obtain the books’ titles of the author Baudelaire</i>
Result	{firstname:'Charles', lastname:'Baudelaire', title:'Les Paradis artificiels'} {firstname:'Charles', lastname:'Baudelaire', title:'Le Spleen de Paris'} {firstname:'Charles', lastname:'Baudelaire', title:'Les fleurs du mal'} {firstname:'Guillaume', lastname:'Apollinaire', title:'Alcools'}	{title:'Paradis artificiels'}
Missing	{firstname:PAUL, lastname:VERLAINE, title:'Les saturnies'}	{title:'Le Spleen de Paris'} {title:'Les fleurs du mal'}
	It is not included in the results because the node’s type (i.e. label) is not Author but rather Writer .	It is not included because labeled as Book

3 Related Work

The problem of querying heterogeneous data is an active research domain studied in several contexts such as data-lake, federated database [15], data integration, schema matching [16]. We classify the state-of-the-art as follows.

Schema Integration. The schema integration process is an intermediary step to facilitate the query execution. In [16] the authors present a survey on techniques used to

automate the schema integration process. The schema integration techniques may lead to data duplication and original structure loss, which affects the support of legacy applications.

Physical Re-factorization. Several works are conducted to enable querying data without any prior schema restrictions. Generally, they propose to flatten data into a relational form [11, 17, 18]. SQL queries are formulated based on relational views built on top of the inferred structures. This strategy suggests performing heavy physical re-factorization. Hence, this process requires additional resources such as external relational database and extra efforts to learn the new relational schema whenever new schemas are inserted.

Schema Discovery. Other works propose to infer implicit schemas. The idea is to give an overview of the different elements present in the integrated data [19, 21]. In [12] the authors propose summarizing all schema under a skeleton to discover the existence of fields or sub-schema. In [20] the authors suggest extracting all structures to help developers while designing their applications. The limitation with such logical view is the need to manual process while building queries by including all attributes and their corresponding paths.

Others works suggest resolving the heterogeneity problem by working on the query side. Query rewriting is a strategy to rewrite an input query into several derivations to overcome the heterogeneity [14, 24, 25]. Most of works are designed in the context of the relational database where heterogeneity is usually restricted to the lexical level only. In NoSQL, the first papers focused on document stores and using another query language that cannot be applied to oriented graph systems [22].

4 EasyGraphQuery: Query Rewriting Engine

EasyGraphQuery differs from the conventional systems, which require a prior knowledge of the different schemas to formulate adequate queries. *EasyGraphQuery* takes for input the user’s query, rewrites it using the dictionary to eventually extract similar attributes and run it into Neo4J. Figure 2 gives an overview of the *EasyGraphQuery* architecture.

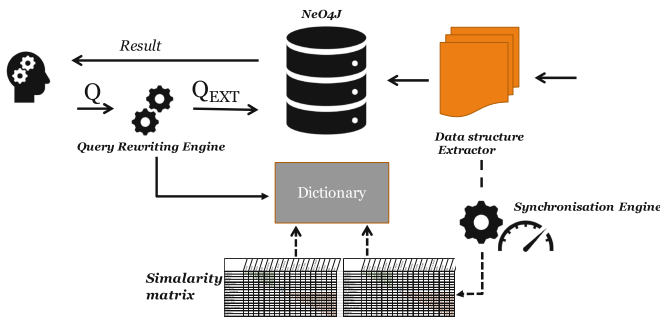


Fig. 2. The *EasyGraphQuery* architecture

The creation of the dictionary is done automatically when inserting data and is updated with each new insertion or update of the already stored data. For performance reasons, the update is continuously operated in the background. By keeping the dates of the last modifications in files, that are independent of the similarity matrix and the dictionary, it is possible to consult the updates status without affecting the queries being executed.

4.1 Data and Dictionary Modeling

Definition 1. A graph G is defined by (V, E, γ)

- $V = \{u_1, \dots, u_N\}$ is the graph's nodes set ;
- $E = \{e_1, \dots, e_M\}$ is the graph's edges set ;
- $\gamma : E \rightarrow V \times V$ is the function that associates each edge with the connected vertices.

We denote $\mathcal{L} = \{l_1, \dots, l_L\}$ a set of terms indicating the different nodes labels and possible relations.

Definition 2. A node u_i is defined as (L_i, S_i)

- $L_i \subseteq \mathcal{L}$ is the set of labels to which the node belongs;
- $S_i = \{a_{i,1}, \dots, a_{i,n_i}\}$ is the node schema composed of a set of attributes

We denote $S_V = \bigcup_{i=1}^{i=N} \left(\bigcup_{l_k \in L_i} \bigcup_{a_{i,j} \in S_i} l_k.a_{i,j} \right)$ the graph node schema.

Definition 3. an edge e_i is defined as $(l_i, S_i, u_{i,1}, u_{i,2})$

- $l_i \in \mathcal{L}$ is the label to which the edge belongs;
- $S_i = \{a_{i,1}, \dots, a_{i,n_i}\}$ is the edge's schema composed of a set of attributes;
- $u_{i,1}$ et $u_{i,2}$ are the source and the target nodes connected by edge; $\gamma(e_i) = \{(u_{i,1}, u_{i,2})\}$.

We denote $S_E = \bigcup_{i=1}^{i=M} \left(\bigcup_{a_{i,j} \in S_i} l_i.a_{i,j} \right)$ the graph edges schema. Thus, $S_G = S_N \cup S_M$ is the graph's attributes schema.

We define $\mathcal{L}_G = \left(\bigcup_{i=1}^{i=N} \mathcal{L}_i \right) \cup \left(\bigcup_{i=1}^{i=M} l_i \right)$.

Example. Let us consider the graph illustrated in Fig. 1.

$V = \{u_1, \dots, u_{13}\}$; $E = \{e_1, \dots, e_9\}$;

- $\gamma = \{(e_1, (u_1, u_5)), (e_2, (u_1, u_6)), (e_3, (u_1, u_7)), (e_4, (u_2, u_8)), (e_5, (u_2, u_9)), (e_6, (u_3, u_{10})), (e_7, (u_4, u_{11})), (e_8, (u_4, u_{12})), (e_9, (u_4, u_{13}))\}$.

Node $u_1 = (\{Author\}, \{firstname, lastname, birth_date, date_of_death\})$ and node $u_7 = (\{Book\}, \{number, title, year\})$. Edge $e_3 = (Book, \{\}, u_1, u_7)$.

In order to consider the different heterogeneity aspects of the graph (structural, syntactic and semantic), we introduce two data dictionaries that allow to determine for each element of the graph (label, attribute), similar elements.

Definition 4. The data dictionary $dict_{label}$ is defined by

$$dict_{label} = \{ (l_i, \nabla_i) \}$$

- $l_i \in \mathcal{L}_G$ is a graph's label;
- $\nabla_i = \cup_{\forall l_j \in \mathcal{L}_G | sim(l_i, l_j) \geq \delta} l_j \subseteq \mathcal{L}_G$ is the set of similar labels. The similarity metric, called *sim*, measures a normalized score ranged in [0..1] expressing the degree of similarity (the more l_i and l_j are similar, the closer the score is to 1). $\delta \in [0..1]$ is the threshold according to which labels l_i and l_j are considered similar.

Definition 5. The data dictionary $dict_{attribut}$ is defined as

$$dict_{attribut} = \{ (l_i \cdot a_{i,k}, \Delta_{i,k}) \}$$

- $l_i \cdot a_{i,k} \in S_G$ is a graph's attribute ;
- $\Delta_{i,k} = \cup_{\forall l_j \cdot a_{j,l} \in S_G | sim(a_{i,k}, a_{j,l}) \geq \delta} l_j \cdot a_{j,l} \subseteq S_G$ is the set of similar attributes.

Similarity is calculated between the eventual heterogeneous elements of the graph. In this paper, we only consider the labels, attributes of nodes and edges. The heterogeneity aspects considered are structural, syntactic and semantic. Two matrices are constructed to determine the similarities between the graph's elements: the syntactic similarity matrix is based on the *Leivenshtein* measure while the semantic similarity matrix is based on the *Lin* measure. We do not detail the pre-processing applied during multi-terms like `To_Write` or `birth_date`, when calculating matrices. We can consider extending the approach with other similarity measures, and improve the process by a weighted combination of these various measures [8, 10].

In this paper, we only consider the attributes structural heterogeneity. The labels structural heterogeneity is not examined. That means that an attribute can be located at various positions in the graph, marked by the labels that prefix the **property**.

4.2 The Algebraic Core of Operators

The interrogation is based on a set of elementary operators forming a closed minimum core. We denote G_{in} the queried graph and G_{out} the resulting graph. These elementary operators allow projection and selection (restriction) operations.

Definition 6. Projection allows reducing the graph to the structural elements specified in the structural pattern along with the projected list of attributes

$$Pattern \pi_{Attribute}(G_{in}) = G_{out}$$

- *Pattern* is a path defined as $l_0-l_1-\dots-l_p$ where every $l_i \in \mathcal{L}_G$ stands for the class of a node or an edge.
- *Attribute* is a set (possibly empty) of attributes $l_i \cdot a_{i,k}$ where $l_i \in Pattern$ and $a_{i,k} \in S_i$.

Definition 7. The selection operator restricts the structures' elements to only those satisfying a selection predicate; we denote:

$$Pattern\sigma_{Predicate}(G_{in}) = G_{out}$$

- *Pattern* is a path defined as $l_0-l_1-\dots-l_p$ where every $l_i \in \mathcal{L}_G$ stands for the class of a node or an edge.
- *Predicate* is a selection condition. A simple predicate is expressed as $l_i.a_{i,k} \omega_k v_k$ where $a_{i,k} \subseteq S_i$ is an attribute, $\omega_k \in \{=;>;<;\neq;\geq;\leq\}$ is a comparison operator, and v_k is a value. Predicates can be combined with operands $\Omega = \{\vee, \wedge, \neg\}$ forming a complex predicate.

The predicates of a complex selection, combining several predicates, are represented in their conjunctive normal form: $Predicate = \bigwedge_x (\bigvee_y p_{x,y})$.

Example. Let us consider the queries from Sect. 2.3. We can express these queries with an algebraic representation (internal) as follows:

Projection. «Get the lastname, the firstname, and the title of the books written by the authors»: $(\text{Author} \rightarrow \pi_{\text{Author.firstname, Author.lastname, l.title}})$.

We present the obtained results in the Table 1. When the attributes are projected, the identifiers of the nodes (u_i) and the edges (e_i) are lost; this breaks the closure principle of the algebraic core, thus not allowing to combine these results with a new operation.

Projection and Selection. «Get the titles of the books for author having name = 'Baudelaire'»

$\text{Author} \rightarrow \pi_{\text{Author.firstname, Author.lastname, l.title}} (\text{Author} \rightarrow \sigma_{\text{Author.firstname='Charles'} \wedge \text{Author.lastname='Baudelaire'}})$

The obtained results are given in the Table 2.

Table 2. Results of the selection and projection operations combination.

G_{out}
firstname:'Charles', lastname:'Baudelaire', title:'Le Spleen de Paris'
firstname:'Charles', lastname:'Baudelaire', title:'Les fleurs du mal'

The use of this internal representation, of the interrogation operations on graphs, does not support the heterogeneity of the graph's elements. Therefore, these queries' results remain incomplete regarding the information represented in the graph.

We present, in the following, the rewriting process of these internal queries allowing to obtain a complete result that is transparent to the user and dynamic (without data transformation).

4.3 Queries Rewriting Algorithm

Neo4J does not provide native operators to manage the graphs heterogeneity; *e.g.*, the `match` operation is very case-sensitive and does not automatically allow comparisons of labels and attributes that are syntactically, semantically or structurally similar. This sensitivity leads to ignore several data that are relevant to the result. Our approach aims at assisting users with querying, by automatic query reformulation. This process makes use of the dictionary and indirectly the similarity matrix to reformulate the query by determining similar elements (nodes, edges, and attributes). The following algorithm describes the automatic rewriting process of the user's query.

The function $exists(A, L)$ verifies if L includes the pattern constituted from the labels resulting from A . The union operation, denoted \cup , allows unifying two graphs

$$G_1 \cup G_2 = G_{out} | V_{out} = V_1 \cup V_2; E_{out} = E_1 \cup E_2; \gamma_{out} : E_{out} \rightarrow V_{out} \times V_{out} | e_{out} \in \gamma_1 \vee e_{out} \in \gamma_2$$

Algorithm : Automatic extension of the user's query

Input : Q

Output : Q_{ext}

begin

$Q_{ext} \leftarrow id$

foreach $q_x \in Q$ **do**

switch q_x **do**

case $Pattern\pi_{Attribute}$ **do** *// projection*

$L_{ext} \leftarrow \prod_{i=1}^p \nabla_i$

$A_{ext} \leftarrow$

$q_{ext} \leftarrow id$

foreach $L \in L_{ext}$ **do**

foreach $A \in A_{ext}$ **do**

if $exists(A, L)$ **then** $q_{ext} \leftarrow q_{ext} \cup L\pi_A$

end

end

end

$Q_{ext} \leftarrow Q_{ext} \circ (q_{ext})$

end

case $Pattern\sigma_{Predicate}$ **do** *// selection*

$L_{ext} \leftarrow \prod_{i=1}^p \nabla_i$

$P_{ext} \leftarrow \wedge_x \left(\vee_y \left(\vee_{a_{l,k} \in \Delta_{x,y}} l_x \cdot a_{l,k} \bar{w}_{l,k} v_{l,k} \right) \right)$

$q_{ext} \leftarrow id$

foreach $L \in L_{ext}$ **do**

foreach $P \in P_{ext}$ **do**

if $exists(P, L)$ **then** $q_{ext} \leftarrow q_{ext} \cup L\sigma_P$

end

end

end

$Q_{ext} \leftarrow Q_{ext} \circ (q_{ext})$

end

end

end

end.

Example. Let us consider the following query

```
Author--1 $\mathcal{T}$ Author.firstname,Author.lastname,l.title (
    Author--1 $\mathcal{O}$ Author.firstname='Charles'.Author.lastname='Baudelaire')
```

The projection operator is rewritten according to the different similar labels of the pattern, $\nabla_{Author} = \{Author, author, Writer\}$ and $\nabla_{Book} = \{Book, book, Publication\}$, and the different similar attributes, $\Delta_{Author.firstname} = \{Author.firstname, Writer.firstname, author.firstname\}$ et $\Delta_{Author.lastname} = \{Author.lastname, Writer.lastname, author.lastname\}$.

The algorithm constructs the following sets, according to which the operator is rewritten.

$$L_{ext} = \{ Author, author, Writer \} \times \{ \} \times \{ Book, book, Publication \}$$

$$= \{ \{Author, , Book\}, \{Author, , book\}, \{Author, , Publication\}, \{Writer, , Book\}, \{Writer, , book\}, \{Writer, , Publication\}, \{author, , Book\}, \{author, , book\}, \{author, , Publication\} \}$$

$$A_{ext} = \{ Author.firstname, Writer.firstname, author.firstname \} \times \{ Author.lastname, Writer.lastname, author.lastname \} \times \{ Book.title, book.title, Publication.title \}$$

$$= \{ \{Author.firstname, Author.lastname, Book.title\}, \dots, \{author.firstname, author.lastname, Publication.title\} \}$$

The selection operator is rewritten according to the different labels of the selection pattern, $\nabla_{Author} = \{Author, author, Writer\}$, and to the different similar attributes of the predicate, $\Delta_{Author.firstname} = \{Author.firstname, Writer.firstname, author.firstname\}$ and $\Delta_{Author.lastname} = \{Author.lastname, Writer.lastname, author.lastname\}$.

Then, the operator is rewritten according to the set constructed by the algorithm.

$$L_{ext} = \{ Author, author, Writer \} \times \{ \} \times \{ Book, book, Publication \} =$$

$$\{ \{Author, , Book\}, \{Author, , book\}, \{Author, , Publication\}, \{Writer, , Book\}, \{Writer, , book\}, \{Writer, , Publication\}, \{author, , Book\}, \{author, , book\}, \{author, , Publication\} \}$$

$$P_{ext} = \{ \{ Author.firstname='Charles', Writer.firstname='Charles', author.firstname='Charles' \}, \{ Author.lastname='Baudelaire', Writer.lastname='Baudelaire', author.lastname='Baudelaire' \} \}$$

5 Experiments

We use the *EasyGraphQuery* tool to evaluate the query rewriting algorithm proposed in this article as well as the construction of the defined dictionary.

Dataset. To validate our approach, we consider ontology data because of their strong structural heterogeneity. We used the *Conference Track* collection made available by OAEI 2017¹. These ontologies lack instances; so we had to generate synthetic instances. The goal is to evaluate the cost of interrogation; the generation and the loading times are not evaluated [26–28].

Tests Environment. We use a cluster composed of a machine (i5-4 core, 8Go RAM, 2To hard drive, 1 Gb/s network) in which we have installed a Neo4J instance – version 3.2.

The Queries Set. We defined a set of 10 queries: three queries for selection, three queries for projection, and four queries to evaluate the selection-projection combination. The set of queries based on the different comparison operators supported by Cypher language. We employed the classical comparison operators, i.e. {<, >, ≤, ≥, =, } for numerical values as well as classical logical operators, i.e. {and, or, exists} between query predicates. Also, we employed a regular expression to deal with string values.

5.1 Setting up the Dictionary and the Similarity Matrix

In these experiments, we study the time needed to create and update the similarity dictionary. Table 3 shows the maintenance time of the dictionary as the ontologies are brought in. The results are clearly influenced by the number of elements (labels, edges, attributes) already present in the graph. Indeed, the log file is regularly analyzed by our parser, but it is not cleaned at each pass.

Table 3. Maintenance time of the syntactical dictionary according to the number of ontologies

Number of ontologies	2	4	6	8
Creation/update time (in seconds)	1.3 s	4.2 s	13.5 s	18.7 s
The dictionary size (KB)	2.7	2.9	3.4	3.5
the parsed logs file size (KB)	1333	14534	17602	21265

5.2 Evaluation of the Query Rewriting Module

In this experiment, we study the additional cost of our proposed approach, i.e. an interrogation with a rewritten query via our similarity algorithm, compared to the cost of a non-rewritten query, called initial query. We also compare the cost of the reformulated query against the sum of the costs of the subqueries, obtained from the decomposition of the reformulated queries.

The Table 4 reports the execution time of the rewritten queries, the initial queries, and the subquery cumulative execution times. A first comparison addresses the execution time of the rewritten queries and the cumulative execution time of the

¹ <http://oaei.ontologymatching.org/2017/>

sub-queries. We can note that the execution time of our approach is, at worst, equal to the cumulative execution time of sub-queries, and it can go up to 2 times faster (in the case of combined queries, for example in the case of our dataset). On the other hand, it is at best equal to the execution time of an initial query.

Table 4. Comparison of the execution time (in seconds) of the reformulated queries and the initial queries (without reformulation)

		Reformulated query	Initial query	Cumulative resulting queries
Projection	Q1.1	316	222	316
	Q1.2	0.160	0.008	0,166
	Q1.3	0.027	0.0013	0.027
Selection	Q2.1	4.05	2.98	4.8
	Q2.2	0.77	0.77	0.85
	Q2.3	2.34	1.73	3.73
Combination (Projection - Selection)	Q3.1	0.2734	0.2082	0.4062
	Q3.2	0.0055	0.0059	0.0073
	Q3.3	0.434	0.0431	0.9342
	Q3.4	0.324	0.324	0.659

To get a deeper understanding of these results, we have plotted the execution of our queries. For example, during the execution of the query Q1.2 where we can notice that during a reformulated query (where our algorithm uses the operator ‘Union’), Neo4J starts the execution of the two ‘Match’ simultaneously; and the union of the two results is consolidated only after the completion of the last ‘Match’ (the one with the most rows). More precisely, in this projection query Q2.1, two types of labels are evaluated: the first corresponds to the label of the initial query, which processes 35054 lines; the second corresponds to the label added by our rewriting algorithm and which deals with 10000 lines. The number of lines explains the results illustrated in Table 2 and shows why our solution is at most equal to the execution time of the slowest sub-query and at best is equal to the initial query.

6 Conclusion

In this paper, we have defined an approach based on the construction of data similarity dictionaries allowing a rewriting of the users’ queries without transforming the stored data. Our method is able to overcome the interrogation problem caused by the data heterogeneity in graph-oriented NoSQL storage systems. Our approach computes for each attribute, the set of its similar attributes (syntactic, semantic, and structural heterogeneity) and it transparently rewrites users’ queries. Rewritten queries allow to enrich initial queries and return the complete set of data.

As a perspective for this work, we intend to extend our mechanisms to support more heterogeneity aspects; for example, consider the entities heterogeneity. We will

also expand the algebraic core of operators supported by the rewrite engine; for example, by integrating aggregation operations.

References

1. Beyer, K.S., Ercegovac, V., Gemulla, R., Balmin, A., Eltabakh, M., Kanne, C.-C., Ozcan, F., Shekita, E.J.: Jaql. In: Proceedings of VLDB Conference (2011)
2. Radic, D.: Influence of schemaless approach on database authorization. In: Hadžikadić, M., Avdaković, S. (eds.) IAT 2017. LNNS, vol. 28, pp. 243–248. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-71321-2_21
3. Floratou, A., Teletia, N., DeWitt, D.J., Patel, J.M., Zhang, D.: Can the elephants handle the NoSQL onslaught? Proc. VLDB Endow. **5**(12), 1712–1723 (2012)
4. Holzschuher, F., Peinl, R.: Performance of graph query languages: comparison of cypher, gremlin and native access in Neo4J. In: EDBT/ICDT (2013)
5. Getoor, L., Machanavajjhala, A.: Entity resolution: theory, practice & open challenges. VLDB Endow. **5**(12), 2018–2019 (2012)
6. Chevalier, M., El Malki, M., Kopluku, A., Teste, O., Tournier, R.: How can we implement a multidimensional data warehouse using NoSQL? In: Hammoudi, S., Maciaszek, L., Teniente, E., Camp, O., Cordeiro, J. (eds.) ICEIS 2015. LNBIP, vol. 241, pp. 108–130. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-29133-8_6
7. Cattell, R.: Scalable SQL and NoSQL data stores. SIGMOD Rec. **39**(4), 12–27 (2011)
8. Megdiche, I., Teste, O., Trojahn dos Santos, C.: An extensible linear approach for holistic ontology matching. In: Groth, P., Simperl, E., Gray, A., Sabou, M., Krötzsch, M., Lecue, F., Flöck, F., Gil, Y. (eds.) ISWC 2016. LNCS, vol. 9981, pp. 393–410. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46523-4_24
9. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity flooding: a versatile graph matching algorithm and its application to schema matching, pp. 117–128 (2002)
10. Shvaiko, P., Euzenat, J.: A Survey of schema-based matching approaches. In: Spaccapietra, S. (ed.) Journal on Data Semantics IV. LNCS, vol. 3730, pp. 146–171. Springer, Heidelberg (2005). https://doi.org/10.1007/11603412_5
11. Tahara, D., Diamond, T., Abadi, D.J.: Sinew: a SQL system for multi-structured data. In: 2014 SIGMOD, pp. 815–826. ACM (2014)
12. Wang, L., Zhang, S., Shi, J., Jiao, L., Hassanzadeh, O., Zou, J., Wangz, C.: Schema management for document stores. Proc. VLDB Endow. **8**(9), 922–933 (2015)
13. Lin, C., Wang, J., Rong, C.: Towards heterogeneous keyword search. In: Proceedings of the ACM Turing 50th Celebration Conference-China, p. 46. ACM (2017)
14. Papakonstantinou, Y., Vassalos, V.: Query rewriting for semistructured data. In: ACM SIGMOD Record, vol. 28, pp. 455–466. ACM (1999)
15. Sheth, A.P., Larson, J.A.: Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Comput. Surv. (CSUR) **22**(3), 183–236 (1990)
16. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB J. **10**(4), 334–350 (2001)
17. Chasseur, C., Li, Y., Patel, J.M.: Enabling JSON document stores in relational systems. In: WebDB, vol. 13, pp. 14–15 (2013)
18. DiScala, M., Abadi, D.J.: Automatic generation of normalized relational schemas from nested keyvalue data. In: Proceedings of the 2016 ICM, pp. 295–310. ACM (2016)
19. Baazizi, M.-A., Lahmar, H.B., Colazzo, D., Ghelli, G., Sartiani, C.: Schema inference for massive JSON datasets. In: EDBT (2017)

20. Herrero, V., Abelló, A., Romero, O.: NOSQL design for analytical workloads: variability matters. In: Comyn-Wattiau, I., Tanaka, K., Song, I.-Y., Yamamoto, S., Saeki, M. (eds.) ER 2016. LNCS, vol. 9974, pp. 50–64. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46397-1_4
21. Sevilla Ruiz, D., Morales, S.F., García Molina, J.: Inferring versioned schemas from NoSQL databases and its applications. In: Johannesson, P., Lee, M.L., Liddle, Stephen W., Opdahl, Andreas L., López, Ó.P. (eds.) ER 2015. LNCS, vol. 9381, pp. 467–480. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25264-3_35
22. Ben Hamadou, H., Ghazzi, F., Péninou, A., Teste, O.: Towards schema-independent querying on document data stores. In: DOLAP 2018 (2018)
23. Clark, J., DeRose, S., et al.: XML path language (XPath) version 1.0 (1999)
24. Boag, S., Chamberlin, D., Fernandez, M.F., Florescu, D., Robie, J., Simeon, J., Stefanescu, M.: XQuery 1.0: an XML query language (2002)
25. Bourhis, P., Reutter, J.L., Suarez, F., Vrgoč, D.: JSON: data model, query languages and schema specification. In: SIGMOD, pp. 123–135. ACM (2017)
26. Chevalier, M., El Malki, M., Kopliku, A., Teste, O., Tournier, R.: Document-oriented data warehouses: models and extended cuboids. In: RCIS 2016, pp. 1–11 (2016)
27. Chevalier, M., Malki, M.E., Kopliku, A., Teste, O., Tournier, R.: implementation of multidimensional databases in column-oriented NoSQL systems. In: Morzy, T., Valduriez, P., Bellatreche, L. (eds.) ADBIS 2015. LNCS, vol. 9282, pp. 79–91. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23135-8_6
28. El Malki, M., Ben Hamadou, H., El Malki, N., Kopliku, A.: MPT: suite tools to support performance tuning in NoSQL systems. In: CEIS 2018 (2018)



Selection of Bitmap Join Index: Approach Based on Minimal Transversals

Issam Ghabry^(✉), Sadok Ben Yahia, and M. Nidhal Jelassi

Faculty of Sciences of Tunis, LIPAH-LR11ES14, El Manar,
University of Tunis El Manar, 2092 Tunis, Tunisia
ghabry.issam@gmail.com, issam.ghabry@fst.utm.tn,
{sadok.benyahia,nidhal.jelassi}@fst.rnu.tn

Abstract. Decision systems handle a large volume of data usually stored in a data warehouse. The latter is modeled by a star schema that typically has a central fact table and a set of dimension tables. The queries corresponding to this type of model are very complex. In order to reduce the cost of running these queries, one common solution would be to ensure a good physical design of data warehouses. In this respect, the binary join indexes (BJI) are very suitable for reducing the cost of running these joins. In this paper, we introduce a BJI selection approach based on a key notion of the theory of hypergraphs, namely minimal transversal. The final configuration obtained is composed of several indexes that optimize the cost of executing of the set of queries. The carried out experiments prove the relevance of our approach versus those of the literature.

Keywords: Data warehouses · Physical design · Bitmap join index
Hypergraph · Minimal transversal

1 Introduction

A data warehouse is a central repository of integrated data from one or more disparate sources. It is a structure (similar to a database) that aims to consolidate business data for analytical purposes and to provide support for strategic decision-making. Roughly speaking, it is a huge pile of information historized, organized, purified, integrated and coming from several sources of data, used for analysis and decision support [11]. In fact, the data warehouse is the best way that researchers and professionals have found to model information for analytical purposes. The data in the warehouse must be organized in a simple and easy way to facilitate their operation as well as their analysis by a decision-maker. Undoubtedly, these analyses require the execution of complex and particular queries, which must result in a representation of the data along several axes of analysis (analysis of habits, trends, purchasing preferences, etc.). The data manipulated in the context of data warehouses is represented in multidimensional form, which is better suited for supporting analysis processes. This model

allows observing the subject (the fact) and the different perspectives of the analysis or measures of interest (dimensions). Given the huge volume of data and the complexity of the queries used, one of the key elements that ensure a good physical design of a data warehouse is to choose the right optimization structures [7]. Each of them is linked to an NP-Complete optimization problem and supported by some approaches proposed by the research community.

Several optimization techniques have been proposed to optimize star join queries. These techniques fall into two categories: (i) redundant optimization techniques such as indexes and materialized views; and (ii) non-redundant optimization techniques such as horizontal fragmentation and parallel processing. The index selection problem has attracted the interest of many researchers. Indeed, it is one of the most important issues in the physical design of databases in general and advanced database applications especially. The problem consists in selecting an appropriate set of indexes for a data warehouse using the queries of the *workload* under some constraints like storage, maintenance, etc. However, the selection of an optimal set of indexes is a very hard problem [6, 8] due to the large data volume and the exponential number of candidate attributes that can be used during the selection process.

In this paper, our main challenge is the optimization of the query response time and we particularly focus on bitmap join indexes, since they are well adapted to data warehouses. The main originality of the introduced approach stands on the use of the minimal transversals to select the final configuration of indexes. Indeed, the data warehouse can be designed as a hypergraph. In this respect, *BJIs* can be seen as the smallest minimal transversal in size terms. It is worth citing that our approach is the first one which uses minimal transversals for selecting a *BJI* configuration.

The remainder of this paper is organized as follows. We introduce some basic notions about *BJI* and hypergraph theory in Sect. 2. In Sect. 3, we present the related work regarding the bitmap join index selection. Then, we detail our approach in Sect. 4 and we discuss related experimental results in Sect. 5. We finally conclude this paper and provide research perspectives in Sect. 6.

2 Key Notions

In the following, we review two key topics related to our proposal, (i) bitmap join index selection problem; and (ii) the hypergraph theory.

2.1 Bitmap Join Index *BJI*

The index selection problem has been shown to be a hard problem [8]. Indeed, choosing the optimal index configuration that can optimize star join queries and reduce their execution time is far from being a trivial task. This index configuration must fulfill certain constraints imposed in a data warehouse such as the storage space, the number of indexes per table, the maintenance and updating time, etc. Therefore, this type of problem is considered in the literature

as a critical issue in physical design since a *BJI* is usually defined on a set of attributes from different tables. To formalize this problem, we can say that in the input, we have three key parameters:

1. A set of queries $Q = \{Q_1, Q_2, \dots, Q_m\}$ that constitutes the *workload*,
2. A data warehouse schema that contains a fact table F and a set of dimension tables D_i ,
3. Maximum storage cost constraints that we do not have to exceed.

Client Table			
RID ^c	CID	Name	Country
6	616	Gilles	France
5	515	Yves	USA
4	414	Patrick	Tunisia
3	313	Didier	Tunisia
2	212	Eric	France
1	111	Pascal	France

Sales Table				
RID ^s	CID	PID	TID	Price
1	616	106	11	25
2	616	106	66	28
3	616	104	33	50
4	515	104	11	10
5	414	105	66	14
6	212	106	55	14
7	111	101	44	20
8	111	101	33	27
9	212	101	11	100
10	313	102	11	200
11	414	102	11	102
12	414	102	55	103

Bitmap Join Index			
RID	France	USA	Tunisia
1	1	0	0
2	1	0	0
3	1	0	0
4	0	1	0
5	0	0	1
6	1	0	0
7	1	0	0
8	1	0	0
9	1	0	0
10	0	0	1
11	0	0	1
12	0	0	1

Fig. 1. An example of BJI

The *BJI* selection problem consists in automatically selecting an appropriate set of *BJIs* that minimize the total query response time and the running cost under resource constraints (storage, maintenance, etc.). Recall that each *BJI* is defined on one or several selection attributes, called *indexable attributes* joining their dimension tables with the fact table using key attributes [15]. The *BJI* selection problem is considered an NP-Complete problem [8]. Therefore, developing an exhaustive algorithm providing an optimal and accurate solution in a reasonable time is infeasible. In this respect, the research community looked for creating heuristic-based algorithms to reach an approximation of the optimal solution represented as an index configuration based on a set of candidate attributes.

To show how to construct a *BJI*, we suppose that we have an attribute A having n distinct values v_1, v_2, \dots, v_n belonging to the dimension table D . We also suppose that we have a fact table F , which is composed of m instances. Figure 1 shows an example of a *BJI* defined between a fact table (Sales) and a dimension table (Client) based on one attribute (Country). Generally, the construction of a *BJI* on attribute A , is as follows:

1. Create n vectors each composed of m entries.
2. The i^{th} bit of the vector corresponding to a value v_k is set to 1 if the tuple of rank i of the fact table is joined with a tuple of the dimension table D , such that the value of A of this tuple is equal to v_k , otherwise it is set to 0.

The binary nature of the *BJI* improves query performance by enabling AND, OR, NOT logical operations, etc. These operations make it possible to search for n-tuples that fulfil conjunctive or disjunctive predicates. *BJI* are especially of benefit for the Count(*) queries, where the access to the *BJI* only allows these queries to be answered. Note that the size of a *BJI* is proportional to the cardinality of the indexed attributes (number of distinct values). Because of this, they are often recommended for low cardinality attributes.

Example 1. To illustrate the use and the interest of bitmap join indexes, let us consider the data of Fig. 1. Suppose that we have a partial schema of a relational data warehouse represented by the dimension table (Client) and one fact table (Sales). Let Q be the query that the administrator would like to optimize:

```
SELECT Count(*)
FROM Client C, SALES S
WHERE C.Country='Tunisia'
AND C.CID=S.CID
```

To do so, we create a bitmap join index Cl-C_BJI between the fact table SALES and the dimension table CLIENT based on Country attribute as follows:

```
CREATE BITMAP INDEX Cl-C_BJI
ON SALES(CLIENT.Country)
FROM SALES S, CLIENT C
WHERE S.CID= C.CID
```

To execute the above query, the query optimizer just accesses the bitmap corresponding to the column representing Tunisia, without joining SALES and CLIENT tables. This example shows the efficiency of bitmap join indexes for executing this kind of queries.

After extracting from a given workload a set of so called indexable attributes, we build a query-attribute matrix. The latter is called a *hypergraph*, the rows represent workload queries are now called *hyperedges* and the set of all the indexable attributes are called *vertices* of the hypergraph. In this paper, we model a given workload of the data warehouse by a hypergraph, on which we apply our *BJI* selection approach. In the following, we recall some key notions borrowed from the hypergraph theory.

2.2 Hypergraph Theory

Hypergraphs generalize the notion of graphs by defining the notion of hyperedges, which can contain one, two or more vertices, unlike classical edges that can join only two vertices [3]. From a theoretical point of view, the hypergraphs make it possible to generalize certain theorems of graphs, even to factorize several of them in one. Hypergraphs are sometimes preferred to graphs because they better model certain types of constraints from a practical point of view. A hypergraph $H = (S, E)$ consists of two sets S and E , and is defined as follows.

Definition 1. *Hypergraph [3]: Let the couple $H = (S, E)$ with $S = \{s_1, s_2, \dots, s_n\}$ a finite set and $E = \{e_1, e_2, \dots, e_m\}$ a family of parts of E . H is a hypergraph on S if:*

1. $e_i \neq \emptyset, i \in \{1, \dots, m\}$
2. $\bigcup_{i=1, \dots, m} e_i = S$

The elements of S are called *vertices* and those of E are called *hyperedges* of the hypergraph.

Example 2. Figure 2 illustrates a hypergraph $H = (S, E)$ such $S = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)$ and $E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$ with $e_1 = \{1, 2, 3\}$, $e_2 = \{2, 4, 5, 6\}$, $e_3 = \{4, 7\}$, $e_4 = \{3, 4, 8\}$, $e_5 = \{8, 9, 10, 11\}$, $e_6 = \{5, 8, 11\}$ and $e_7 = \{3, 6, 7, 8\}$.

A transversal of a hypergraph H is a set of vertices $s \subseteq S$ that intersects each hyperedge of H at least once.

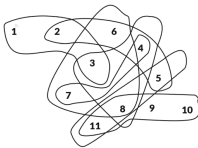


Fig. 2. Hypergraph H

{3, 4, 8}	{3, 4, 11}	{2, 7, 8}	{2, 4, 8}
{1, 4, 8}	{3, 6, 7, 8}	{3, 6, 7, 11}	{3, 5, 7, 8}
{3, 5, 7, 9}	{3, 5, 7, 10}	{3, 5, 7, 11}	{3, 4, 5, 10}
{3, 4, 5, 9}	{2, 3, 7, 11}	{2, 4, 6, 11}	{2, 4, 7, 11}
{1, 6, 7, 8}	{1, 5, 7, 8}	{1, 4, 6, 11}	{1, 4, 7, 11}
{2, 4, 5, 6, 10}	{2, 4, 5, 7, 10}	{2, 4, 5, 6, 9}	{2, 4, 5, 7, 9}
{1, 4, 5, 6, 10}	{1, 4, 5, 7, 10}	{1, 4, 5, 6, 9}	{1, 4, 5, 7, 9}

Fig. 3. Minimal transversals of H

Definition 2. *Minimal transversal [3]: Let a hypergraph $H = (S, E)$. The set of transversals of H , denoted $\gamma(\mathcal{H})$, is equal to: $\gamma(\mathcal{H}) = \{T \subset S \mid T \cap s_i \neq \emptyset, \forall i = 1, \dots, |E|\}$. A transversal T of $\gamma(\mathcal{H})$ is minimal if there is not another transversal R of $\gamma(H)$ included in T : $R \in \gamma(\mathcal{H})$ s.t. $R \subset T$.*

We denote $(\mathcal{M}_{\mathcal{H}})$, the set of minimal traversals defined on H .

From a hypergraph $H = (S, E)$, the set of minimal traversals $(\mathcal{M}_{\mathcal{H}})$ allows the construction of the transversal hypergraph, which is a hypergraph where all its hyperedges are minimal transversals.

Definition 3. *Given a hypergraph $H = (S, E)$, the minimum number of vertices of a transversal set is called the transversality number of the hypergraph H and is designated by: $\tau(\mathcal{H}) = \min\{|T|, \text{ such as } T \in (\mathcal{M}_{\mathcal{H}})\}$ [3].*

Example 3. Thus, in the illustrative example of Fig. 2, the hypergraph has twenty eight minimal transversals, they are plotted in Fig. 3. The transversality number of the hypergraph H is equal to 3 since the smallest minimal transversal of $\mathcal{M}_{\mathcal{H}}$ is composed of 3 vertices, and the set $(\mathcal{M}_{\mathcal{H}})$ of the hypergraph is equal to: $\{\{1, 4, 8\}, \{2, 4, 8\}, \{2, 7, 8\}, \{3, 4, 8\}$ et $\{3, 4, 11\}\}$. The determination of the transversality number appears in many combinatorial problems [10].

3 Related Work

Determining the candidate attributes to build a *BJI* configuration for a data warehouse is a combinatorial problem and the corresponding search space is huge. Consequently implementing exhaustive or enumerative solutions is simply infeasible. To reduce this cost, we pass as the first step in a pruning phase to reduce the search space. Several works have focused on this phase, which relies on certain selection criteria that favors attributes over others such as the frequency of appearances of the attributes, and the size of tables, etc. [9]. However, selecting an optimal set of indexes is a very difficult problem [6], because of the large amount of data and the exponential number of candidate attributes that can be used in the selection process. To reduce this complexity, most of the proposed approaches [1, 2, 5, 6, 14, 18, 19] prune the search space to reduce the number of candidate attributes by using several techniques. The indexes are selected with different algorithms such as greedy algorithms, data mining algorithms, linear algorithms, etc. Even a genetic algorithm has been proposed for the selection of *BJI* [4]. In the remainder of this paper, we focus on the selection of *BJI* in data warehouses modeled by a star schema.

Aouiche et al. [1] proposed a data mining based approach (DM) by using the CLOSE algorithm [17] to mine frequent closed itemsets. The latter are used to prune the *BJI* search space. An itemset is closed if it has no superset with the same *support*. The support is the number of transactions of the workload that contain an attribute X , is called the *absolute support* of X whereas the fraction of transactions is called its relative support (both denoted by $sup(X)$). Thus, an itemset is frequent when $sup(X)$ reaches at least a user-specified minimum threshold called *minsup*. The generated itemsets are used to build a set of *BJI* candidates. A greedy algorithm, based on the cost model, is then executed to select a final set of *BJI*. This model estimates the data access cost using the indexes and their maintenance cost expressed in the number of input/output operations (I/Os). The authors consider the access frequencies of the attributes as only one criterion for generating frequent closed itemsets.

Bellatreche et al. [2] proposed the DYNACLOSE algorithm. The latter is an adaptation of the data mining approach by injecting pruning parameters, which allows reaching the elected bitmap join index. The proposed algorithm relies on the penalty of the frequent attributes defined on tables of small sizes. This approach starts by determining the indexable attributes. An indexable attribute is an attribute that is not a key attribute of the dimension table nor it does not belong to the fact table. After the determination of the context matrix, this approach computes the support for each indexable attribute and provides the closed frequent itemsets. And finally, it computes the value of the FITNESS function of these itemsets as follows: for a frequent itemset m ,

$$Fitness(m) = \frac{1}{n} \times \left(\sum_{j=1}^n (\alpha_j \times sup_j) \right). \quad (1)$$

where n is the number of non-key attributes j in m , j is the attribute in the itemset m to test. sup_j represents the support of j , α_j is a penalization parameter determined by the equation: $\alpha_j = \frac{|D_j|}{|F|}$; the notations $|D_j|$ and $|F|$ respectively denote the size of the dimension table D to which the attribute j belongs and the fact table F in number of pages.

After the determination of the frequent itemsets, the DYNACLOSE approach is used in the pruning phase of the *BJI*. For example, a frequent itemset that does not contain any non-key attributes of the dimension table will be removed. Purification generates a set of candidate indexable attributes. This set is defined by the union of non-key attributes belonging to the frequent itemsets generated. To get a better result, the approaches of the literature, recommend the use of alternative parameters, such as the size of tables, the length of each tuple, the size of disk page, etc.

To overcome the limitations of these approaches, we introduce a new algorithm, called MT-BJI, which uses the minimal transversals of a hypergraph and takes into account the above mentioned parameters as part of the pruning metric. Indeed, the cost of join operations heavily depends on the size of joined tables. Once the minimal transversals are computed and the pruning phase is processed, the selected *MT*s allows building the set of *BJIs*.

4 Our Approach MT-BJI

In this section, we thoroughly describe the MT-BJI algorithm. The main thrust of the latter lies in the use of minimal transversal of a hypergraph. Indeed, each hyperedge represents a query and each vertex represents an attribute that has been the subject of a selection or projection predicate.

4.1 Bitmap Join Index Selection Strategy

In order to select a set of bitmap join indexes minimizing the overall query execution time, we consider an approach subdivided into 5 steps. The latter are illustrated in Fig. 4. In the following, we describe in detail the steps of our *BJI* selection strategy.

Step 1: Workload Analysis: SQL queries in the workload are processed by an automatic parser to extract all attributes that might be considerate as indexes, called *indexable attributes*. These attributes are those present in the *Where* clauses of the considered queries.

Step 2: Construction of the hypergraph: from the extracted attributes in the previous step, we build a matrix (queries-attributes) called *hypergraph*, such that into the hyperedges stands for the queries of the workload and into the vertices stands for the attributes to index.

Step 3: Extraction of the minimal transversals: once the hypergraph is built, we delve into the process of the minimal transversals extraction. For this step, we take advantage of the recent works of the literature dedicated

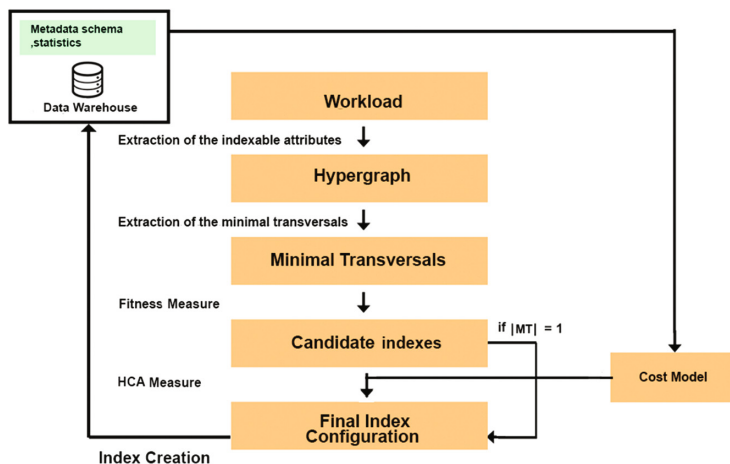


Fig. 4. MT-BJI approach

to the computation of the minimal transversals. Therefore, we use the best existing algorithm in terms of performance, i.e., SHD of Murakami and Uno [13] for computing MT s. Our choice is solely dictated by the interesting processing times of the latter. We automatically configured the SHD algorithm to only generate the smallest MT s through the GETMINTRANSVERSALITY function introduced by Jelassi et al. [12], which returns the transversality number of a hypergraph.

Step 4: Construction of the set of candidate indexes: after the generation of the set of minimal transversals, a phase of generation of candidate indexes is necessary. Thus, we proceed as follows:

1. *Applying the FITNESS measure:* Instead of relying on the frequency of occurrence presented by the support as a criterion for determining frequent itemsets, we modified the approach of Bellatreche et al. [2] that relies on the FITNESS measure. The latter penalizes each MT , which contains attributes belonging to small dimension tables, taking into account the cardinalities of the dimension tables and the size of the system page.

If after running the FITNESS measure, several MT s are retained in this case we use an additional metric to choose the best possible configuration.

2. *Elimination of High Cardinality Attributes HCA:* The main cause of the explosion of the BJI size is the cardinality of the indexable attributes. In this measure, the attributes are sorted in a descending order with respect to their cardinality. Next, we compute, for each retained MT , the sum of the cardinality of their attributes and we only keep the MT having the smallest total in terms of the sum of cardinality.

Step 5: Construction of the final configuration: once all the previous steps are completed, only the selected MT will be used to build the final configuration of indexes after eliminating vertices that represent non-indexable attributes, i.e., key attributes or attributes belonging to the fact table.

In our work, the cost model is used to predict the input/output cost of each attribute of the minimal transversal and to choose the best attributes to build the *BJI*. As our working hypothesis is to choose the *BJI* configuration without considering any constraint, then all the indexable attributes of the resulting minimal transversal could be of use to build the *BJI* configuration. The cost model remains an important tool if we want to add or remove indexes from the final configuration to satisfy some constraints.

4.2 The MT-BJI Algorithm

The MT-BJI algorithm whose pseudo code is given in Algorithm 1, inputs a hypergraph and outputs a *BJI* configuration. After the construction of the hypergraph through the attributes able to act as indexes from the workload, the process implemented by our approach begins with the computation of the transversality number (line 1), which is equal to the cardinal of the smallest minimal transversal. Subsequently, it connects by the extraction of all the minimal transversals that have a size equal to the transversality number (line 2). Once it has all *MTs*, it computes for each one the FITNESS measure (line 4) as follows : for a minimal transversal *mt*,

$$Fitness(mt) = \sum_{i=1}^n (sup_i \times \alpha_i) \quad (2)$$

where *n* is the number of non-key attributes of the minimal transversal *mt*, *i* is an attribute of *mt*. *sup_i* represents the support of *i*, α_i is a penalization parameter equal to: $\alpha_j = \frac{|D_j|}{|F|}$. The notations $|D_j|$ and $|F|$ respectively denote the size of the dimension table *D* to which the attribute *i* belongs and the fact table *F* in number of pages.

Algorithm 1. MT-BJI Algorithm

Input: Hypergraph $H = (S, E)$

Output: Final configuration : minimal transversal.

```

1  $t := \text{GETMINTRANSVERSALITY}(H)$ .
2  $MT := \text{SHD}(S, E, t)$ .
3 foreach  $mt \in MT$  do
4    $\lfloor \text{FITNESS}(mt)$ ;
5  $f := \max(\text{FITNESS}(MT))$ .
6 if  $|f| > 1$  then
7   foreach  $mt \in f$  do
8      $\lfloor \text{HCA}(mt)$ ;
9   return  $\min(\text{HCA}(f))$ 
10 Else return  $(f)$ .
```

After computing the FITNESS measure, we only keep the *MT*s having the maximum value. If only one minimal transversal maximizes the FITNESS measure, then it will be retained as the final configuration. In the case of a single *MT* is obtained, then we compute the sum of cardinalities of each attribute of the *MT*s, by using the cardinality measure *HCA* (line 8). The *MT* retained is that which minimizes the *HCA* measure (line 9). Since the main cause of the explosion of the *BJI* size is the cardinality of the indexed attributes.

4.3 A Study Case

We consider the following example, with a star schema containing two dimensions tables CHANNELS (denoted by *Ch*), CUSTOMERS (denoted by *C*) and a fact table SALES (denoted by *S*). The cardinalities of these tables (number of instances) are given in Table 1. The respective sizes of the considered instances of CUSTOMERS, CHANNELS, and SALES is 24, 24, and 36, and the page size $PS = 65536$.

Table 1. Table respective cardinalities

Table	Cardinality
CHANNELS	5
CUSTOMERS	50000
SALES	16260336

Assume that the five queries are executed most often on the corresponding data cube (c.f., Table 2). To make it easier to construct the context matrix, we rename the indexable attributes as follows: SALES: *cust_id* = A_1 , CUSTOMERS: *cust_id* = A_2 ; CUSTOMERS: *cust_gender* = A_3 , CHANNELS: *channel_id* = A_4 , SALES: *channel_id* = A_5 ; CHANNELS: *channel_desc* = A_6 . The matrix is given by Fig. 5.

Table 2. The test queries

(1) select S.channel_id, sum(S.quantity_sold) from S, C where S.channel_id=C.channel_id and C.channel_desc='Internet' group by S.channel_id
(2) select S.channel_id, sum(S.quantity_sold), sum(S.amount_sold) from S, C where S.channel_id=C.channel_id and C.channel_desc='Catalog' group by S.channel_id
(3) select S.channel_id, sum(S.quantity_sold), sum(S.amount_sold) from S, C where S.channel_id=C.channel_id and C.channel_desc='Partners' group by S.channel_id
(4) select S.cust_id, avg(quantity_sold) from S, C where S.cust_id=C.cust_id and C.cust_gender='M' group by S.cust_id
(5) select S.cust_id, avg(quantity_sold) from S, C where S.cust_id=C.cust_id and C.cust_gender='F' group by S.cust_id

	A ₁	A ₂	A ₃	A ₄	A ₅	A ₆
Q ₁	0	0	0	1	1	1
Q ₂	0	0	0	1	1	1
Q ₃	0	0	0	1	1	1
Q ₄	1	1	1	0	0	0
Q ₅	1	1	1	0	0	0

Fig. 5. Our study case matrix

Once the construction of the hypergraph is complete, the process put in place by our approach starts by computing the transversality number. In this example, the transversality number is equal to 2. Therefore, only the minimal transversals having a size equal to the transversality number are retained. In the considered hypergraph we mine 9 *MTs* : {(1,4), (1,5), (1,6), (2,4), (2,5), (2,6), (3,4), (3,5), (3,6)}. As soon as the algorithm finishes the generation of all the *MTs* of size 2, it computes for each of them the corresponding *FITNESS* value. It is worth mentioning that only the attributes *A₃* and *A₆*, i.e., respectively *cust_gender* and *channel_desc*, are indexable attributes. Since the attributes, *A₁*, *A₂*, *A₄* and *A₅* are non-indexable ones because *A₁* and *A₅* belong to the fact table *SALES* and *A₂* and *A₄* are respectively key attributes of the *CUSTOMERS* and *CHANNELS* tables. From our trivial example, only 4 of the 9 *MTs* will be considered since they contain indexable attributes which are given in Table 3.

We apply the *FITNESS* measure for the minimal transversal *MT₂* {3,4} as an example, we solely consider the attribute *A₃* since *A₄* is a key one :

$$\begin{aligned}
 \text{FITNESS}(MT_2) &= \text{sup}(A_3) \times \alpha_3 \\
 &= \frac{2}{5} \times \frac{\lceil \frac{50000 \times 241}{65536} \rceil}{\lceil \frac{1626033 \times 36}{65536} \rceil} \\
 &= \frac{2}{5} \times \frac{19}{894} \\
 &= 0.0085
 \end{aligned}$$

And likewise for all remaining *MTs*. The results of the *FITNESS* measure of all the *MTs* are given in Table 3, and according to these last results we will retain only *MT₄* {3,6} by what it has the greatest value. Since after applying the *FITNESS* measure only one *MT* has been selected, then we do not need to use the cardinality measure(*HCA*).

Table 3. Fitness measure values

MTs	Attributes	Fitness
<i>MT₁</i>	{A ₁ , A ₆ }	0.0001
<i>MT₂</i>	{A ₃ , A ₄ }	0.0085
<i>MT₃</i>	{A ₃ , A ₅ }	0.0085
<i>MT₄</i>	{A ₃ , A ₆ }	0.0086

If the minimal transversal MT_4 did not exist, the algorithm will return MT_2 and MT_3 and in this case it automatically passes to the computation of cardinality of each transversal. In this scenario, MT_2 has a total of 16310336. The latter is the sum of the cardinalities of the tables to which belongs the attributes i.e., SALES and CUSTOMERS. These cardinalities are given in Table 1. MT_3 has a total of 50005 and according to these values, the algorithm will return MT_3 as a final configuration.

5 Experimental Study

In order to show the relevance and the efficiency of our bitmap join index selection strategy, we have run tests on a data warehouse implemented within Oracle 12c, on a *i7* machine 2.4 GHz with a 8 GB main memory. Our experimental study is carried out on the data warehouses resulting from the two benchmarks TPC-H¹ 1 GB and SSB 1 GB [16]. Recall that the TPC-H benchmark includes a fact table *LineItem* and 7 dimension tables (*Customers*, *Nation*, *Orders*, *Part*, *PartSupp*, *Region*, *Supplier*). The proposed benchmark also includes 22 OLAP queries. The SSB benchmark is composed of a fact table *Lineorder* and 4 dimension tables (*Date*, *Supplier*, *Customer*, *Part*). We also consider a workload composed of 30 decision-support queries containing several joins [2]. The benchmarks were stored and the queries execution were done using the DBMS *Oracle 12c*.

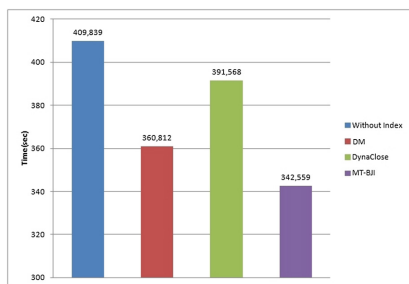


Fig. 6. SSB workload execution

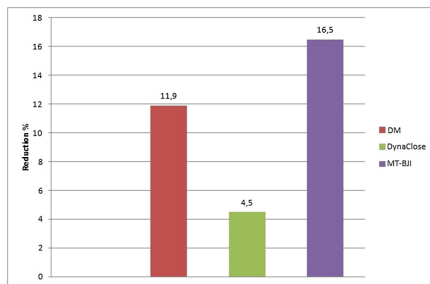


Fig. 7. SSB reduction rate

The experiments were carried out in four steps: (i) identification of the value of *minsup* which gives a significant number of *FCI* for the approaches (DM, DYNACLOSE); (ii) evaluation of the running time by executing the queries without considering any storage constraint; (iii) evaluating the different approaches (DM, DYNACLOSE, MT-BJI) in terms of running time without considering any storage constraint; and (iv) the computation of the theoretical input/output cost.

¹ <http://www.tpc.org/tpch/default.asp>.

After having executed all the queries of the workload, the histogram in Fig. 6 shows the execution time of the 30 queries of the benchmark SSB. Figure 6 shows that MT-BJI decreases the execution time and provides better results compared than other approaches. This is predictable in the fact that MT-BJI improves all the queries of the workload since the chosen attributes belong to a minimal transversal and appear in every query at least once. The histogram in Fig. 7 shows the total reduction rate of each approach, MT-BJI has the highest gain of 16.5 % for DM and DYNACLOSE is 11.9 % and 4.5 % respectively.

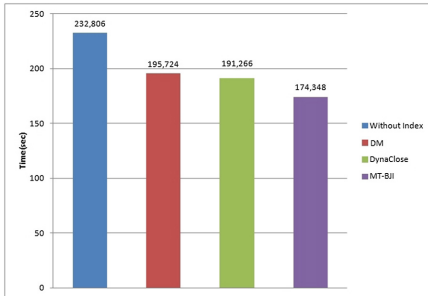


Fig. 8. TPC-H workload execution

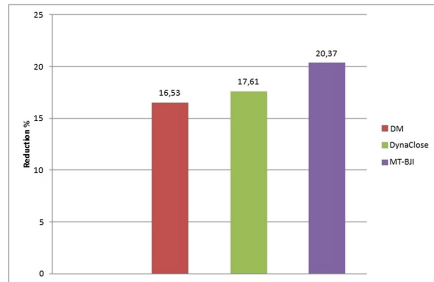


Fig. 9. TPC-H reduction rate

We have conducted another bunch of experiments using this time the benchmark TPC-H. The obtained results are plotted in Fig. 8. These results show that our approach MT-BJI outperforms the other ones since it reduces the execution time and ameliorates the performance. Figure 9 summarizes the obtained results and provides the total reduction rate of each approach. The best gain in response time is obtained by our approach outperforming DM and DYNACLOSE. The obtained results confirm the utility of using the minimal transversals in *BJI* selection.

Figures 10 and 11 put the light on how different indexing approaches reduce the cost of running queries. To quantify the theoretical cost we used the cost model introduced by Aouiche et al. [1]. The main result is that MT-BJI overrides all approaches, since it reduces the cost of execution in terms of inputs/outputs (I/O) operations, from 24.8 million I/O operations to only 4.7 million for the benchmark SSB, which remains lower than those of DM and DYNACLOSE. Likewise for the benchmark TPC-H. Owing to the developed cost model, we were able to quantify the quality of the selected optimization structure *BJI* on the attributes proposed by the different approaches. This model estimates data access cost through indexes. Thus, we can see that it is more beneficial to consider the use of minimal transversals to select *BJI*s configuration.

Based on the results of the experimental study, our approach gives better performance in terms of running time as well as the running cost. Consequently, we can say that the use of the minimal transversals is a good solution for solving the *BJI* selection problem. To sum up, the use of minimal transversals for the

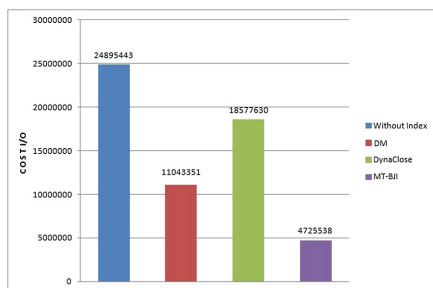


Fig. 10. SSB queries cost

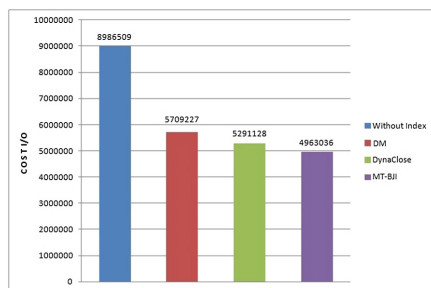


Fig. 11. TPC-H queries cost

selection of *BJI* allows us to have a high speed and efficiency in the extraction of indexable attributes and to reduce the risk of neglecting interesting candidate attributes. Our approach ameliorates all the queries of the workload and does not require a prior configuration by the administrator, while the other approaches require to fix the *minsup* beforehand.

6 Conclusion

BJIs are redundant optimizing structures that pre-compute star joins. These indexes are characterized by a binary representation allowing the use of logical operations to evaluate conjunctions or disjunctions of predicates contained in star join queries. They are generally recommended for low cardinality attributes.

In this paper, we were interested in the problem of selecting a *BJI* configuration and we formalized the selection of *BJI* as an optimization problem. We have shown the main limitations of previous approaches and proposed a new strategy for bitmap join index selection in data warehouses. This strategy first exploits minimal transversals of a hypergraph obtained by the SHD algorithm from a given workload to build a set of candidate bitmap join indexes to optimize the selection and generate the appropriate set of *BJIs*. Our work shows that the idea of using the hypergraph theory for the physical design of a data warehouse is a promising approach.

In the future, we aim to improve our approach by taking into account the cost of updating the *BJIs* as an additional selection constraint. Since the selection of *BJI* depends on a workload set at the input, studying the influence of the evolution of this workload on the performance of the obtained configuration is necessary. This study will provide a dynamic approach to *BJI* selection. We also aim to propose an energy-aware *BJI* selection approach in which we consider the energy consumption as an additional constraint.

References

1. Aouiche, K., Darmont, J., Boussaïd, O., Bentayeb, F.: Automatic selection of bitmap join indexes in data warehouses. In: Tjoa, A.M., Trujillo, J. (eds.) *DaWaK 2005*. LNCS, vol. 3589, pp. 64–73. Springer, Heidelberg (2005). <https://doi.org/10.1007/11546849-7>
2. Bellatreche, L., Missaoui, R., Necir, H., Drias, H.: A data mining approach for selecting bitmap join indices. *J. Comput. Sci. Eng.* **2**(1), 206–223 (2008)
3. Berge, C.: *Hypergraphs: Combinatorics of Finite Sets*, 3rd edn. North-Holland, New York (1989)
4. Bouchakri, R., Bellatreche, L., Hidouci, K.-W.: Static and incremental selection of multi-table indexes for very large join queries. In: Morzy, T., Härder, T., Wrembel, R. (eds.) *ADBIS 2012*. LNCS, vol. 7503, pp. 43–56. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33074-2_4
5. Chaudhuri, S., Narasayya, V.: An efficient cost-driven index selection tool for Microsoft SQL server. In: *International Conference on Very Large Databases*, pp. 146–155 (1997)
6. Chaudhuri, S., Datar, M., Narasayya, V.: Index selection for databases: a hardness study and a principled heuristic solution. *IEEE Trans. Knowl. Data Eng.* **16**(11), 1313–1323 (2004)
7. Chaudhuri, S., Narasayya, V.: Selftuning database systems : a decade of progress. In: *VLDB*, pp. 3–14 (2007)
8. Comer, D.: The difficulty of optimum index selection. *ACM Trans. Database Syst. (TODS)* **3**(4), 440–445 (1978)
9. Golfarelli, M., Rizzi, E., Saltarelli, S.: Index selection for data warehousing. In: *Proceedings 4th International Workshop on Design and Management of Data Warehouses (DMDW 2002)*, Toronto, Canada, pp. 33–42 (2002)
10. Hagen, M.: *Algorithmic and computational complexity issues of MONET*. Ph.D Dissertation. Institut für Informatik, Friedrich-Schiller-Universität Jena (2008)
11. Inmon, W.H.: *Building the Data Warehouse*, 3rd edn. Wiley, Hoboken (2002)
12. Jelassi, M.N., Langeron, C., Ben Yahia, S.: Efficient unveiling of multi-members in a social network. *J. Syst. Softw.* **94**, 30–38 (2014)
13. Murakami, K., Uno, T.: Efficient algorithms for dualizing large-scale hypergraphs. In: *Proceedings of the 15th Meeting on Algorithm Engineering and Experiments (ALENEX 13)*, New Orleans, USA, p. 113 (2013)
14. Necir, H., Drias, H.: A distributed maximal frequent itemset mining with multi agents system on bitmap join indexes selection. *Int. J. Inf. Technol. Manag.* **14**(2/3), 201–214 (2015)
15. *Oracle Data Warehousing Guide*, Oracle 11g documentation library (2007)
16. O’Neil, P., O’Neil, B., Chen, X.: *Star Schema Benchmark Revision 3*, June 2009
17. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Discovering frequent closed itemsets for association rules. In: Beeri, C., Buneman, P. (eds.) *ICDT 1999*. LNCS, vol. 1540, pp. 398–416. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-49257-7_25
18. Toumi, L., Moussaoui, A., Ugur, A.: A linear programming approach for bitmap join indexes selection in data warehouses. *Procedia Comput. Sci.* **52**, 169–177 (2015)
19. Valentin, G., Zuliani, M., Zilio, D.C., Lohman, G.M. and Skelley, A.: DB2 advisor: an optimizer smart enough to recommend its own indexes. In: *ICDE 2000*, pp. 101–110 (2000)



Scalable Random Sampling K-Prototypes Using Spark

Mohamed Aymen Ben HajKacem^(✉), Chiheb-Eddine Ben N'cir,
and Nadia Essoussi

LARODEC, Université de Tunis, Institut Supérieur de Gestion de Tunis,
41 Avenue de la liberté, cité Bouchoucha, 2000 Le Bardo, Tunisia
medaymen.hajkacem@gmail.com,
{chiheb.benncir,nadia.essoussi}@isg.rnu.tn

Abstract. Big data clustering has become an important challenge in machine learning. Several Big data frameworks have been developed to scale clustering methods for Big data analysis. One such framework called Spark works well for iterative algorithms by supporting in-memory computations. We propose in this paper a new Scalable Random Sampling K-Prototypes, implemented on Spark framework. This method is able to perform grouping from mixed large scale data. Experiments realized on simulated and real data sets show the efficiency of the proposed method compared to existing k-prototypes methods.

Keywords: Big data · K-prototypes · Spark · Sampling

1 Introduction

Large volume of data are being collected everyday given the increasing involvement of humans in the digital space. Such huge amount of data containing useful information which is called Big data. Several definitions of Big data currently exist [3]. For example, the most used definition of Big data is characterized by three Vs, namely, Volume, Variety and Velocity. Volume refers to the large scale data. Variety indicates the different data types and formats like numeric, categorical and text. Velocity refers to the speed at which data is generated [3]. Hence, analyzing and organizing such data using machine learning techniques has become an important challenge in Big data analysis.

Clustering is a promising machine learning technique, which has been used to organize data into groups of similar data points called also clusters. Over the past decades, several clustering methods have been designed based on various theories and approaches [4]. These clustering methods were well used in several applications such as intrusion detection, customer segmentation, document clustering and image organization. However, conventional clustering methods fail to perform grouping from Big data. For example, k-prototypes clustering [5], one of the most popular method to cluster mixed data, does not scale with huge volume

of data [9]. This is explained by the high computational cost of this method to build the grouping from large amount of data.

To deal with this issue, we propose in this paper a new **Scalable Random Sampling K-Prototypes**, implemented on an Apache Spark framework, referred to as SRS-KP. In our implementation, we aim to exploit the flexibility provided by Spark, by using in-memory operations that reduce the running time of existing MapReduce solutions. Furthermore, we randomly select a small number of data points and approximate the cluster centers from such data. Therefore, the proposed method requires computation of only small portion of the whole data set, leading to a significant speedup of existing k-prototypes methods.

The remainder of this paper is organized as follows: Sect. 2 provides related works which propose to deal with mixed and large scale data. Then, Sect. 3 presents the Big data frameworks used in this work and the k-prototypes. After that, Sect. 4 describes the proposed SRS-KP method while Sect. 5 presents experiments that we have performed to evaluate the efficiency of the proposed method. Finally, Sect. 6 presents conclusion and future works.

2 Related Works

To deal with large scale data, several clustering methods which are based on parallel frameworks have been designed in the literature [6, 8, 13]. Most of these methods use the MapReduce framework. For instance, Zhao et al. [13] implemented k-means through MapReduce framework. This method first assigns each data point to the nearest cluster prototypes in the map function. The reduce function then updates the new cluster prototypes. Then, this method iterates calling the two functions several times until convergence. Ludwig [6] proposed the parallelization of fuzzy c-means clustering using MapReduce framework. This method is based consist of two MaReduce jobs. The first MapReduce job calculates the cluster centers by scanning the input data set. The second MapReduce job also iterates over the data set and calculates the distances to be used to update the membership matrix as well as to calculate the fitness. Although the later discussed methods offer for users an efficient analysis of large scale data using MapReduce, they can not support the mixed types of data and are limited to only numeric attributes. However, they usually employ a pre-processing step based on transforming categorical attributes to numeric values. Then, they apply a numeric distance measure for computing similarity between data objects. However, these transformation strategies are often time consuming and produce information loss, leading to inaccurate clustering results.

To deal with mixed large scale data, Ben HajKacem et al. [1] have proposed a parallelization of k-prototypes method through MapReduce framework. This method first assigns each data point to the nearest cluster centers in the map phase. The reduce phase then updates the new cluster centers. Then, this method iterates calling the two phases several times until convergence. Despite its efficiency to cluster large scale mixed data, this method has a shortcoming related to the inherent conflict between MapReduce and k-prototypes. K-prototypes is

an iterative algorithm which requires to perform some iterations for producing optimal results. In contrast, MapReduce has a significant problem with iterative algorithms. As a consequence, the whole data set must be loaded from the file system into the main memory at each iteration. Then, after it is processed, the output must be written to the file system again. Therefore, many of I/O disk operations occur during each iteration and this decelerates the running time. To overcome this problem, we propose the Scalable Random Sampling with k-prototypes for handling Big data.

3 Background

This section first presents the Big data frameworks used in this work, then presents the k-prototypes method.

3.1 MapReduce Framework: Hadoop and Spark

MapReduce [2] is a parallel programming framework used to process large scale data across cluster of machines. It is characterized by its highly transparency for programmers to parallelize algorithms in easy and comfortable way. MapReduce is based on two phases namely *map* and *reduce*. Each phase has $\langle key/value \rangle$ pairs as input and output. Apache Hadoop is the most popular implementation of MapReduce for Big data processing and storage on commodity hardware. The use of this framework has become widespread in many fields because of its performance, open source nature, installation facilities and its distributed file system named Hadoop Distributed File System (HDFS). In spite of its great popularity, Hadoop MapReduce has a significant problems with iterative algorithms [7].

Spark [12] is a parallel framework for large scale data processing designed to solve the MapReduce's limitations. It was introduced as part of the Hadoop and it is designed to run with Hadoop, specially by reading data from HDFS. Spark is based on Resilient Distributed Datasets (RDDs), a special type of data structure used to parallelize computations in a transparent way. These parallel structures persist, reuse and cache results in memory. Moreover, Spark provides set of in-memory operators, beyond the standard MapReduce, with the aim of processing data more rapidly on distributed environments. Spark is faster up to 100x than MapReduce.

3.2 K-Prototypes Method

Given a data set $X = \{x_1 \dots x_n\}$ containing n data points, described by m_r numeric attributes and m_t categorical attributes, the aim of k-prototypes [5] is to find k clusters by minimizing the following objective function:

$$J = \sum_{i=1}^n \sum_{j=1}^k u_{ij} d(x_i, c_j), \quad (1)$$

where $u_{ij} \in \{0, 1\}$ is an element of the partition matrix $U_{n \times k}$ indicating the membership of data point i in cluster j and $d(x_i, c_j)$ is the dissimilarity measure which is defined as follows:

$$d(x_i, c_j) = \sum_{r=1}^{m_r} \sqrt{(x_{ir} - c_{jr})^2} + \sum_{t=1}^{m_t} \delta(x_{it}, c_{jt}), \quad (2)$$

where x_{ir} represents the value of numeric attribute r and x_{it} represents the value of categorical attribute t . c_{jr} represents the mean for the all the data points that are present in the cluster j , which can be calculated as follows:

$$c_{jr} = \frac{\sum_{i=1}^{|c_j|} x_{ir}}{|c_j|}, \quad (3)$$

where $|c_j|$ the number of data points assigned to cluster j . c_{jt} represents the most common value (mode) for categorical attributes t and cluster j , which can be calculated as follows:

$$c_{jt} = a_t^h, \quad (4)$$

where

$$f(a_t^h) \geq f(a_t^z), \quad \forall z, \quad 1 \leq z \leq m_c, \quad (5)$$

where $a_t^z \in \{a_t^1 \dots a_t^{m_c}\}$ is the categorical value z and m_c is the number of categories of categorical attribute t . $f(a_t^z) = |\{x_{it} = a_t^z\}|$ is the frequency count of attribute value a_t^z . For categorical attributes, $\delta(p, q) = 0$ when $p = q$ and $\delta(p, q) = 1$ when $p \neq q$. The main algorithm of k-prototypes is described by Algorithm 1.1.

Algorithm 1.1. The main algorithm of the k-prototypes method

Input: X : Data set, k : number of clusters

Output: Cluster centers

begin

1. Choose k initial cluster centers randomly from X .
2. Assign each data point in X to the closest center by computing distances using Equation 2.
3. Update the cluster centers using Equations 3 and 4.
4. If the new cluster centers and the previous ones are the same, then terminate; otherwise, return to Step 2.

end

4 Scalable Random Sampling K-Prototypes Method Using Spark (SRS-KP)

The proposed method consists of two MapReduce jobs namely; *Data sampling* and *Data clustering*. The first MapReduce job is devoted to build a data-sample from the input data. The second MapReduce job is concerned with clustering of the data-sample in order to look for cluster centers.

4.1 Data Sampling MapReduce Job

We generate in this MapReduce job a data-sample from the input data using reservoir random sampling [10]. This section first presents the reservoir random sampling followed by the parallel implementation.

Reservoir Random Sampling: Reservoir random sampling [10] is a technique for selecting random data-sample of fixed size without replacement from an input data set. Let S denotes a data-sample of size r . Initially, the reservoir random sampling algorithm places all the data points in the data-sample until the size becomes r . After that, each data point x_i is accepted for inclusion in the data-sample with the probability of r/i and an accepted data point replaces a randomly selected data point in the data-sample.

Parallel Implementation: The data sampling MapReduce job first creates an RDD object with the input data X formed by m chunks. Then, each chunk is processed in map phase to generate the intermediate data-samples. After that, the reduce phase collects the set of intermediate data-samples in order to generate the final data-sample.

During the map phase, the map function picks a chunk and applies the reservoir random sampling algorithm in order to build an intermediate data-samples of size r . Then, the map function emits the intermediate data-sample as output to a single reduce phase. The reduce phase collects the intermediate data-samples produced in map phase in order to build the final data-sample.

4.2 Data Clustering MapReduce Job

Once the data-sample is built, we propose a second MapReduce job to look for cluster centers. This job first creates an RDD object with the data-sample S formed by m chunks. Then, each chunk is processed in map phase to extract the intermediate centers. After that, the reduce phase processes the set of intermediate centers in order to generate the final cluster centers.

During the map phase, the map function picks a chunk, executes the k-prototypes on that chunk and extracts the cluster centers. In order to obtain a good quality, we record for each extracted center a weight which represents the number of assigned data points. That is to say, we extract from each chunk, k centers and the number of data points assigned to each center. The number of assigned data points to each cluster center represents the importance of that center. Finally, the map function emits the extracted intermediate centers and their weights as the output to a single reduce phase. In our implementation in Spark, we use the `mapPartitions(func)` transformation on each chunk of the RDD separately.

The reduce phase takes the set of intermediate centers and their weights, executes again the k-prototypes algorithm on them and returns the final centers as the output. To ease the implementation of this idea, we use the transformation

ReduceByKey(func) from Spark. It is important to note that we must extend the k-prototypes to take into account the weighted data points when clustering the set of intermediate centers. In order to consider the weighted data points, we must change centers update. If we take into account w_i as the weight of data point x_i , center of a final cluster must be calculated for numeric and categorical attributes using the following equations.

$$c_{jr} = \frac{\sum_{i=1}^{|c_j|} x_{ir} * w_i}{|c_j|}. \quad (6)$$

$$c_{jt} = a_t^h, \quad (7)$$

where

$$f(a_t^h) * w_i \geq f(a_t^z) * w_i, \quad \forall z, \quad 1 \leq z \leq m_c, \quad (8)$$

5 Experiments and Results

5.1 Methodology

In order to evaluate the efficiency of SRS-KP method, we performed experiments on both simulated and real data sets. We tried in this section to figure out the following two points: (i) How much is the efficiency of SRS-KP method when applied to mixed large scale data compared to existing methods? (ii) How the Spark framework can enhance the scalability of the proposed method when dealing with mixed large scale data?

5.2 Environment and Data Sets

The experiments are performed on a cluster of 4 machines where each machine has 1-core 2.30 GHz CPU E5400 and 1 GB of memory. The experiments are executed using Apache Spark version 2.0.2, Apache Hadoop 2.6.0 and Ubuntu 14.04. We conducted the experiments on simulated and real data sets. For simulated data sets, we generate four series of mixed large data sets. These data sets range from 10 millions to 40 millions data points. The numeric values are generated with gaussian distribution. The mean is 350 and the sigma is 100. The categorical values are generated using the data generator developed in¹. In order to simplify the names of simulated data sets, we will use the notations Sim10M, Sim20M, Sim30M and Sim40M to denote a simulated data set containing 10, 20, 30 and 40 millions data points respectively. Concerning real data sets, we use the KDD Cup data set (KDD)² and Poker data set³. Statistics of simulated and real data sets are summarized in Table 1.

¹ <https://projets.pasteur.fr/projects/rap-r/wiki/SyntheticDataGeneration>.

² <https://archive.ics.uci.edu/ml/datasets/KDD+Cup+1999+Data>.

³ <http://archive.ics.uci.edu/ml/datasets/Poker>.

Table 1. Summary of the data sets

Data set	Number of data points	Number of attributes	Domain
Sim10M	10.000.000	10 (5 Numeric, 5 Categorical)	Simulated
Sim20M	20.000.000	10 (5 Numeric, 5 Categorical)	Simulated
Sim30M	30.000.000	10 (5 Numeric, 5 Categorical)	Simulated
Sim40M	40.000.000	10 (5 Numeric, 5 Categorical)	Simulated
KDD	4.898.431	36 (33 Numeric, 3 Categorical)	Intrusion detection
Poker	1.000.000	10 (5 Numeric, 5 Categorical)	Game

5.3 Evaluation Measures

In order to evaluate the quality of the proposed method, we use Sum Squared Error (SSE) [4]. It is one of the most common partitional clustering criteria which aims to measure the squared errors between each data point and the cluster center to which the data point belongs to. SSE can be defined by:

$$SSE = \sum_{i=1}^n \sum_{j=1}^k d(c_j, x_i), \quad (9)$$

where x_i the data point and c_j the cluster center.

In order to evaluate the ability of the proposed method to scale with large data sets, we use in our experiments the Speedup and Scaleup measures [11]. The Speedup measures the ability of the designed parallel method to scale well when the number of machines increases and the size of data is fix. This measure is calculated as follows:

$$Speedup = \frac{T_1}{T_m}, \quad (10)$$

where T_1 the running time of processing data on 1 machine and T_m the running time of processing data on m machines in the used cluster. The Scaleup measures the ability of the designed parallel method to perform well when the number of machines and the size of data set increase simultaneity. The Scaleup is defined by:

$$Scaleup = \frac{T_{n_1}}{T_{h*n_h}}, \quad (11)$$

where T_{n_1} the running time of processing data with size of n on 1 machine and T_{h*n_h} the running time of processing data with size of $h * n$ on h machines.

5.4 Comparison of the Performance of Proposed Methods Versus Existing Methods

We evaluate in this section the efficiency of the proposed method for simulated and real data sets. We fix the number of clusters to 10 for each data set.

The results are reported in Table 2 and the best values are bolded. The obtained results show that SRS-KP method always finish several times faster than k-prototypes and MR-KP methods. For example, Table 2 shows that SRS-KP is found to be 14 times faster than k-prototypes and 4 times faster than MR-KP for Poker data set. In addition, we can observe that more than 95% of the running time was spent in the map phase and this shows that SRS-KP method is truly executed in in-memory. Concerning the quality, SRS-KP method converges to nearly same results of the conventional k-prototypes. However, MR-KP method produce the same SSE values compared to k-prototypes.

Table 2. Comparison of the results with existing methods on simulated and real data sets

Data set	Method	Running time (seconds)	SSE
Sim10M	k-prototypes	2360.28	8.58E+09
	MR-KP	644.88	8.58E+09
	SRS-KP	41.80	9.07E+09
Sim20M	k-prototypes	4887.28	1.73E+10
	MR-KP	1320.88	1.73E+10
	SRS-KP	72.45	1.83E+10
Sim30M	k-prototypes	7776.05	2.57E+10
	MR-KP	2172.08	2.57E+10
	SRS-KP	111.11	2.81E+10
Sim40M	k-prototypes	9669.55	3.39E+10
	MR-KP	2627.59	3.39E+10
	SRS-KP	157.93	3.83E+10
KDD	k-prototypes	984.95	1.22E+10
	MR-KP	246.37	1.22E+10
	SRS-KP	27.69	1.08E+10
Poker	k-prototypes	127.90	1.39E+07
	MR-KP	31.97	1.39E+07
	SRS-KP	09.00	1.80E+07

5.5 Scalability Analysis

We first evaluate in this section the scalability of the proposed method when the number of machines increases using speedup measure. Figure 1 illustrates the speedup results on simulated data sets. For evaluating the speedup values, we first executed the proposed method using just a single machine and then we added additional machines. The obtained results show that our method has a good Speedup when the number of machines is increased. Furthermore, we

can observe that the Speedup becomes nearly linear when we increase the size of data. For example, SRS-KP is 3.61 times faster with 4 machines than single machine for Sim10M data set while it becomes 3.89 times faster with 4 machines for Sim40M.

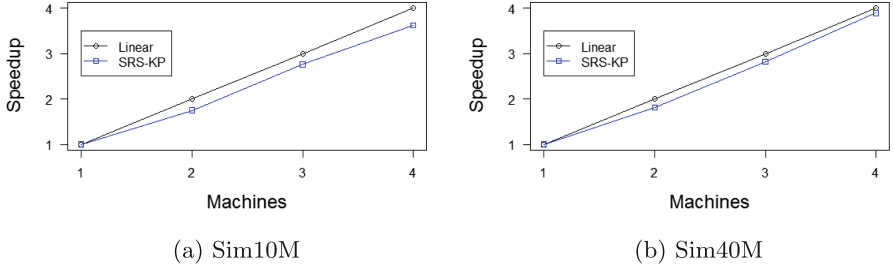


Fig. 1. Speedup results of SRS-KP on Sim10M and Sim40M data sets

Then, we evaluate the scalability of the proposed method when we increase the data set sizes with direct proportion to the number of machines. To do so we execute Sim10M, Sim20M, Sim30M and Sim40M data sets on 1, 2, 3 and 4 machines respectively. Figure 2 illustrates the scaleup results on simulated data sets. From this Figure, we can observe that the proposed method scales very well. The scaleup results have almost a constant value between 0.8 and 0.78. Hence, we can conclude that the performance of the proposed methods is approximately similar when we increase the data set sizes and the number of machines with the same proportion.

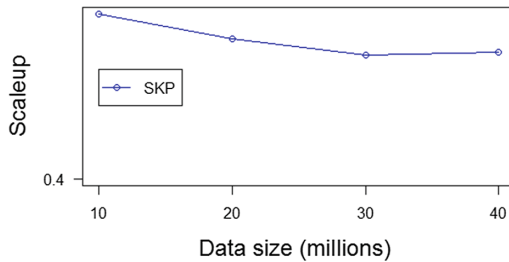


Fig. 2. Scaleup results of SRS-KP on simulated data sets

6 Conclusion

In this paper, we have proposed a Scalable Random Sampling k-prototypes method for handling Big data. The proposed method is implemented on Apache Spark to deal with large scale of mixed data. We show empirically that the

proposed method is (i) efficient in terms of running time, (ii) scalable when the number of machines increases and (iii) able to produce comparable clustering results as existing k-prototypes methods. In the future, we plan to analytically investigate the sample complexity of the proposed method, i.e. the minimum data-sample size required to produce similar clustering results as the conventional k-prototypes method.

References

1. Ben Haj Kacem, M.A., Ben N'cir, C.E., Essoussi, N.: MapReduce-based k-prototypes clustering method for big data. In: Proceedings of Data Science and Advanced Analytics, pp. 1–7 (2015)
2. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. *Commun. ACM* **51**(1), 107–113 (2008)
3. Gandomi, A., Haider, M.: Beyond the hype: big data concepts, methods, and analytics. *Int. J. Inf. Manag.* **35**(2), 137–144 (2015)
4. Han, J., Pei, J., Kamber, M.: *Data Mining: Concepts and Techniques*. Elsevier, Amsterdam (2011)
5. Huang, Z.: Clustering large data sets with mixed numeric and categorical values. In: Proceedings of the 1st Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 21–34 (1997)
6. Ludwig, S.A.: MapReduce-based fuzzy c-means clustering algorithm: implementation and scalability. *Int. J. Mach. Learn. Cybern.* **923–934**, 1–12 (2015)
7. Singh, D., Reddy, C.K.: A survey on platforms for big data analytics. *J. Big Data* **2**(1), 8 (2015)
8. Shahrivari, S., Jalili, S.: Single-pass and linear-time k-means clustering based on MapReduce. *Inf. Syst.* **60**, 1–12 (2016)
9. Vattani, A.: K-means requires exponentially many iterations even in the plane. *Discret. Comput. Geom.* **45**(4), 596–616 (2011)
10. Vitter, J.S.: Random sampling with a reservoir. *ACM Trans. Math. Softw.* **11**(1), 37–57 (1985)
11. Xu, X., Jäger, J., Kriegel, H.P.: A fast parallel clustering algorithm for large spatial databases. In: Guo, Y., Grossman, R. (eds.) *High Performance Data Mining*, pp. 263–290. Springer, Boston (2002). https://doi.org/10.1007/0-306-47011-X_3
12. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets. *HotCloud* **10**(10–10), 95 (2010)
13. Zhao, W., Ma, H., He, Q.: Parallel K-Means clustering based on mapreduce. In: Jaatun, M.G., Zhao, G., Rong, C. (eds.) *CloudCom 2009*. LNCS, vol. 5931, pp. 674–679. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10665-1_71

Data Mining



ERAPN, an Algorithm for Extraction Positive and Negative Association Rules in Big Data

Parfait Bemarisika^{1,2(✉)} and André Totohasina¹

¹ Laboratoire de Mathématiques et d'Informatique ENSET, Université d'Antsiranana, Antsiranana, Madagascar

bemarisikap7@yahoo.fr, andre.totohasina@gmail.com

² Laboratoire d'Informatique et de Mathématiques EA2525, Université de La Réunion, Saint-Denis, France

Abstract. Over the past two decades, the extraction of positive association rules is a significant research area in Big Data. While the negative association rules has been received a lot of attention, they remained in the shadows due to the difficulty of their extraction. In this paper, we propose ERAPN, an algorithm for extraction of valid positive and negative association rules. Frequent patterns can be derived in a single pass to the database, because, of the new technique support counting, called *reduction-access-database*. As for the generation of potential valid association rules, we introduce a new technique, called *reduction-space-rules*, by dividing the space candidates into two. Only half of the candidates have to be studied through this technique. Some experiments will be leaded into such reference databases to complete our study.

Keywords: Big data · Association rules · Reduction-access-data
Reduction-space-rules · ERAPN algorithm

1 Introduction and Motivations

In recent decades, the extraction of association rules [1] is one of the most popular techniques in Big Data. An association rule is a quasi-implication of the form “if **Condition** then **Result**”. Apriori algorithm [2] is the first model that deals with this topic. Despite its notable contribution, Apriori is limited in the extraction of classical (or positive) association rules. therefore, outreach works has been proposed. Brin et al. [6] propose a model generating negative association rules by using the χ^2 measure. Although effective, the model suffers from a lack of memory space due to the lack of use of this χ^2 measure. In [11], the authors present an approach based on knowledge of the field of study. However, this approach is difficult to adapt to the user's problem because of the dependency in the field of study. Wu et al. [15] propose an approach using interest-CPIR pair. One of the key problems lies in pruning: no optimized techniques are used, and

the reresearch space can be exhaustively explored. In [7], the authors propose the PNAR algorithm. Although obtaining notable contributions, PNAR suffers from the high volume of results, due to the lack of use the pair support-confidence. Wilhelmiina proposes the Kingfisher algorithm [14] using the Fisher test. A notable limitation of this model lies in the computation of p -value imposing the exhaustive passes over the database. Guillaume and Papon [10] propose the RAPN algorithm, derived from [2, 7], based on the couple support-confidence and M_G measure (M_{GK} [9] modified). Although effective, RAPN suffers from the relatively high computational cost for the space frequent patterns.

We note from this quick overview state of the art that the extraction of positive and negative association rules remains a major challenge in large databases. Although negative rules have obvious advantages [4, 6], they remain less explored compared to positive rules. We believe that one of the major disadvantages lies in their difficult extraction, this type increases the costs of calculation exponentially. Besides, the current approaches are limited to classical data structure of the Apriori [2] and classical quality measures such as support, confidence and χ^2 . While, this classical data structure imposes the repetitive accesses over database, increasing the computational costs exponentially. Then, the computation of χ^2 on its contingency table is exponential. The couple support-confidence is interpretable: (i) the extraction of frequent itemsets is often tricky in the presence of dense and big data; (ii) the number of rules that can be reduced nevertheless remains high that many prove uninteresting handicapping the expertise of the user. In order to exceed these limits, we propose ERAPN, an algorithm for extraction of frequent patterns and valid association rules with the new couple *support- M_{GK}* . We introduce a new method of the support count, called *reduction-access-database*, based on the new data structure MATRIXSUPPORT and the concept of generator itemsets. Therefore, a simple path of the database allows us to extract all frequent itemsets. As for the generation of valid association rules, we introduce a new technique of selection, noted *reduction-space-rules*, which divides the candidate space into two. Only half of the candidates are to study thanks to this technique. We present the experimental evaluation conducted with databases from the literature by showing performances compared to semantically close algorithms (RAPN [10], WU [15]).

The rest of this paper is organized as follows. Section 2 present the notation, the problem statement and the motivation from support- M_{GK} . Section 3 details the ERAPN algorithm. This section describes our methods for mining frequent patterns and potential valid association rules in big data. Section 4 summarizes our experimental results. A conclusion and outlook are given in Sect. 5.

2 Preliminary Concepts

In this section, we describe the basic concepts from itemset mining (Subsect. 2.1), the limits of couple support-confidence (Subsect. 2.2), and the valid association rules using to new couple support- M_{GK} (Subsect. 2.3).

2.1 Basic Concepts from Itemset Mining

We place in a transactional context which is a triple $\mathcal{B} = (\mathcal{T}, \mathcal{I}, \mathcal{R})$, where \mathcal{T}, \mathcal{I} and \mathcal{R} are finite and not empty sets. An element of \mathcal{I} is called item (or pattern, or attribute), an element of \mathcal{T} is called transaction (or object) represented by a TID-Transaction Identifier, and \mathcal{R} is a binary relationship between \mathcal{T} and \mathcal{I} . However, $|\mathcal{T}|$ and $|\mathcal{I}|$ denotes the total number of transactions and items respectively. The table below represents an example. Given $X, Y \subseteq \mathcal{I}$, $\neg X = \overline{X} = \mathcal{I} \setminus X = \{t \in \mathcal{T} | \exists i \in X : (i, t) \notin \mathcal{R}\}$ is called the logical negation of X . A k -itemset is an itemset of length k . We will use the correspondances (Galois connections [8]) $g(\mathcal{I}) = \{t \in \mathcal{T} | \forall i \in \mathcal{I}, i\mathcal{R}t\}$ and $f(\mathcal{T}) = \{i \in \mathcal{I} | \forall t \in \mathcal{T}, i\mathcal{R}t\}$. The function g is anti-monotony: for all $X, Y \subseteq \mathcal{I}$, if $X \subseteq Y$ then $g(Y) \subseteq g(X)$. It is clear that if $X \subseteq Y$ then $supp(X) \geq supp(Y)$. The applications $\gamma = f \circ g$ and $\gamma' = g \circ f$ are Galois closure operators. An itemset X is closed if $X = \gamma(X)$. A positive rule is a quasi-implication of the form $X \rightarrow Y$. It is called negative rule which consider the absence of the item, i.e., $X \rightarrow \overline{Y}$, $\overline{X} \rightarrow Y$ and $\overline{X} \rightarrow \overline{Y}$, where $X \cap Y = \emptyset$. X is called the premise and Y the conclusion. To determine an association rule interesting, two measures are used, support and confidence [1]. The support of X is the number of transactions that contain X , defined as $supp(X) = \frac{|\{t \in \mathcal{T} | X \subseteq t\}|}{|\mathcal{T}|} = \frac{|g(X)|}{|\mathcal{T}|}$, where, for any finite set A , $|A|$ denotes the number of its elements. Denoting by P the intuitive probability measure defined on $(\mathcal{T}, \mathcal{P}(\mathcal{T}))$ by $P(E) = \frac{|E|}{|\mathcal{T}|}$ for $E \subseteq \mathcal{T}$, the support of X can be written in terms of P as $supp(X) = P(X)$. The item X is frequent if its support exceeds a minimum support threshold value, $minsup \in [0, 1]$, i.e. $supp(X) \geq minsup$. The set of all frequent itemsets of a context \mathcal{B} is noted \mathcal{F} set, formally:

$$\mathcal{F} = \{X \subseteq \mathcal{I} | X \neq \emptyset \wedge supp(X) \geq minsup\}. \tag{1}$$

The support and confidence of $X \rightarrow Y$ are defined $supp(X \cup Y) = supp(XY) = \frac{|g(XY)|}{|\mathcal{T}|}$ and $conf(X \rightarrow Y) = P(Y|X) = \frac{supp(XY)}{supp(X)}$. According to Morgan, we have $supp(\overline{X}) = 1 - supp(X)$, $supp(X\overline{Y}) = supp(X) - supp(XY)$, $supp(\overline{X}Y) = supp(Y) - supp(XY)$ and $supp(\overline{X}\overline{Y}) = 1 - supp(X) - supp(Y) + supp(XY)$.

Table 1. Example of transactional context \mathcal{B}

TID	Items	Positive and negative items	Equivalent binary
1	ACD	A¬BCD¬E	10110
2	BCE	¬ABC¬DE	01101
3	ABCE	ABC¬DE	11101
4	BE	¬AB¬C¬DE	01001
5	ABCE	ABC¬DE	11101
6	BCE	¬ABC¬DE	01101

2.2 Limits of the Classical Couple Support-Confidence

Despite its notable contribution, the couple support-confidence selects the uninteresting rules. The examples in Table 2 illustrate this. In this case, the first four columns give the characteristics of the purchase of products A and B, the last four indicate those of the purchase of coffee and tea. We obtain $supp(A \cup B) = 0.72$ and $P(B|A) = 0.9$. These reasonably high values lead us to believe that the persons buying A also buy B. However, we find that the confidence is equal to the probability of the conclusion regardless of the premise (i.e. $P(B|A) = P(B)$), it is a stochastic independence between A and B. The rule $A \rightarrow B$ that seemed interesting is therefore misleading. On the other hand, we obtain $supp(tea \cup coffee) = 0.2$, which assumes that tea favors coffee ($P(coffee|tea) > P(coffee)$). However, the share of people buying coffee regardless of whether they also buy tea is higher, it is a negative dependence between tea and coffee. The rule $tea \rightarrow coffee$ that seemed interesting is therefore misleading. That's why the couple support-confidence extracts uninteresting rules. Therefore, we use an efficient quality measure, M_{GK} [9, 13, 15].

Table 2. Limit of the couple support-confidence

	A	$\neg A$	Σ		coffee	\neg coffee	Σ
B	72	18	90	tea	20	5	25
$\neg B$	8	2	10	\neg tea	70	5	75
Σ	80	20	100	Σ	90	10	100

2.3 Valid Association Rules Using New Couple Support- M_{GK}

Given $X, Y \subseteq \mathcal{I}$, such that $X \cap Y = \emptyset$, the M_{GK} [13] of $X \rightarrow Y$ is defined by:

$$M_{GK}(X \rightarrow Y) = \begin{cases} \frac{P(Y|X) - P(Y)}{1 - P(Y)}, & \text{si } P(Y|X) > P(Y) \text{ (X favors Y), } P(Y) \neq 1 \\ \frac{P(Y|X) - P(Y)}{P(Y)}, & \text{si } P(Y|X) < P(Y) \text{ (X disfav. Y), } P(Y) \neq 0 \end{cases}$$

The measure M_{GK} varies in $[-1, 1]$ while reflecting reference situations such as incompatibility, independence, dependence, and logical implication. Two zones are present: *attractive zone* and *repulsive zone*. The first is a zone that ranges from independence ($P(Y|X) = P(Y)$) to logical implication ($P(Y|X) = 1$), where X favors Y. The second is a zone that ranges from incompatibility ($P(Y|X) < P(Y)$) to independence, where X disfavors Y. The following definition defines the interesting/non-interesting association rules in our approach.

Definition 1. Given $X, Y \subseteq \mathcal{I}$, an association rule $X \rightarrow Y$ is interesting if $P(Y|X) > P(Y)$. It is not interesting if $P(Y|X) \leq P(Y)$ (or it belongs to the attractive zone, but rather far from the logical implication).

If $M_{GK}(X \rightarrow Y) = -1$, then X and Y are incompatible. This corresponds to the repulsion limit between X and Y but to the attraction of couples (X, \bar{Y}) , (\bar{X}, Y) , (Y, \bar{X}) and (\bar{Y}, X) . If $M_{GK}(X \rightarrow Y) = 0$, then X and Y are stochastically independent, it corresponds $X \rightarrow Y$ is not interesting. If $-1 \leq M_{GK}(X \rightarrow Y) < 0$, Y is negatively dependent on X , it corresponds $X \rightarrow Y$ is uninteresting. If $0 < M_{GK}(X \rightarrow Y) \leq 1$, then Y is positively dependent on X , it corresponds $X \rightarrow Y$ could be interesting. If $M_{GK}(X \rightarrow Y) = 1$, then X and Y are strongly correlated, which denotes the logical implication between X and Y . It corresponds that $X \rightarrow Y$ is a rule exact, so it is potentially interesting. Let $minsup$ and $minmgk$ be two minimum thresholds of support and M_{GK} , defined in $[0, 1]$. The rule $X \rightarrow Y$ is said to be valid according to our approach if its support $supp(X \cup Y)$ is frequent and $M_{GK}(X \rightarrow Y) \geq minmgk$. The set of all valid association rules from \mathcal{B} is denoted \mathcal{E}_{RAPN} , formally:

$$\mathcal{E}_{RAPN} = \{X, Y \in \mathcal{I} | supp(X \cup Y) \geq minsup \ \& \ M_{GK}(X \rightarrow Y) \geq minmgk\}. \quad (2)$$

With the same example of Table 2, suppose $minsup = 0.1$ and $minmgk = 0.8$. Because, $M_{GK}(A \rightarrow B) = 0 < 0.8$, moreover A and B are stochastically independent, the association rule $A \rightarrow B$ is invalid, it is not added in \mathcal{E}_{RAPN} . But, $supp(\bar{A} \cup B) = supp(B) - supp(A \cup B) = 0.90 - 0.72 = 0.18 > 0.1$ and $M_{GK}(\bar{A} \rightarrow B) = 0.88 > 0.8$, the rule $\bar{A} \rightarrow B$ is valid, it added in \mathcal{E}_{RAPN} . On the other hand, one has $M_{GK}(tea \rightarrow coffee) < 0$, coffee is negatively dependant on tea. This is a situation we should consider the negative association rules.

3 ERAPN Algorithm

ERAPN is part of a double problematic: the extraction of frequent patterns (Algorithm 1) and valid association rules (Algorithm 3). The first problem is often complex (at worst, it reaches $2^{|\mathcal{I}|}$) and dramatic when one considers the negative items. With the small database in Table 1, we have 1024 different items instead of 32 positive. The same is true for the second problem (at a m -itemset, we have at worst cases $5^m - 2(3^m) + 1$ instead of $3^m - 2^{m+1} + 1$). From Table 1, there are 2640 different rules instead of 180 classic rules. In these dimensions, it is necessary to select only a part, without loss of informations. In [3, 5], we have initiated the solution to these problems, which will be refined in Subjects 3.1 and 3.2 in terms of precision and simplicity.

3.1 Extraction of Frequent Patterns

We show that a single path in the database allows to extract all frequent itemsets using new method, *reduction-access-database*. As noted, the reresearch of frequent space is often complex. The worst case concerns the generation of inferior itemsets (1 and 2-itemsets). To answer this, we develop the new data structure MATRIXSUPPORT (see Table 3). It is a projection of the \mathcal{B} database in relation to its attributes. To each attribute corresponds a cell of the matrix to which

we associate the absolute support, noted $|v_{ij}|$, expressing the number of item v_j appears with the item v_i , where i (resp. j) denotes the i -th line (resp. j -th column) of table. This is used to identify the relative support, $|v_{ij}|/|\mathcal{B}|$. However, the supports will be obtained a simple scan of \mathcal{B} : $supp(A) = |v_{11}|/6 = 3/6$, $supp(AB) = |v_{12}|/6 = 2/6$ et $supp(BC) = |v_{23}|/6 = 4/6$.

Table 3. Formalism of the MATRICESUPPORT in dataset \mathcal{B}

TID	Attributes
1	ACD
2	BCE
3	ABCE
4	BE
5	ABCE
6	BCE

Scan the database \mathcal{B}

MATRICESUPPORT					
$i \setminus j$	A	B	C	D	E
A	3	2	3	1	2
B	-	5	4	0	5
C	-	-	5	1	4
D	-	-	-	1	0
E	-	-	-	-	5

Next, the generation of k -itemsets candidate is obtained from frequent $(k - 1)$ -itemsets. In this case, the supports will be calculated using generator concepts. An itemset X is a generator if it's minimal (of set inclusion) in its equivalence class. The equivalence class of X is given by $[X] = \{X' \subseteq \mathcal{I} | \gamma(X') = \gamma(X)\}$. Note that the calculation of closures is very exponential. However, the following lemma exploits the monotony of support upon set inclusion.

Lemma 1. $\forall X, Y \in \mathcal{I}$, if $X \subseteq Y$ and $supp(X) = supp(Y)$, then $\gamma(X) = \gamma(Y)$.

For example, from the Table 3, $supp(AB) = supp(ABC) = supp(ABE) = supp(ABCE) = 2/6$, we have $\gamma(AB) = \gamma(ABC) = \gamma(ABE) = \gamma(ABCE)$. Therefore, AB is minimal, then it is generator itemset. If the candidate is a not generator, it will be calculated using the following Proposition 1.

Proposition 1. For all X non generator, $supp(X) = \min\{supp(X') | X' \subset X\}$.

Proof. Let \mathcal{I} be a set items. Let X and X_1 be items on \mathcal{I} such that $X_1 \subseteq X$. Due to the growing support, we have $supp(X) \leq supp(X_1)$. In addition (by assumption), X is non-generator, so there is a X' of \mathcal{I} such that $supp(X') = supp(X)$. However, $supp(X_1)$ is minimal in \mathcal{I} , so $supp(X_1) < supp(X')$. Finally, $supp(X) = supp(X_1) = \min\{supp(X') | X' \subset X\}$. □

The support of a non generator k size is exactly the smallest support of its $(k - 1)$ -subsets. For example, from the Table 3, we have $supp(AC) = supp(A) = 3/6$, therefore AC is not generator itemset. However, the superset of its subset ABC is then obtained by $supp(ABC) = \min\{2/6, 3/6, 4/6\} = 2/6$. The following properties generalizes this observation.

Property 1. Given $X \subseteq \mathcal{I}$, if X is a generator, then $\forall X_1 \subseteq X$, X_1 is a generator, whereas if X is not a generator, $\forall X_2 \supseteq X$, X_2 is not a generator.

Theorem 1. Any subset of a generator itemset must also be a generator. Any superset of a nongenerator itemset must also be nongenerator.

Proof. Let X_1 and X_2 be items on \mathcal{I} satisfy $X_1 \subset X_2$. There is a Y non-empty and disjoint of X_1 such as $X_2 = X_1 \cup Y_1$. If X_1 is assumed to be a non-generator, then X_1 admits a proper subset Y_2 that is equivalent to it: $Y_2 \subset X_1$ and $Y_2 \approx X_1$, giving $Y_2 \cup Y_1 \approx X_1 \cup Y_1$. By hypothesis, $X_1 \cap Y_1 = \emptyset$, so $Y_2 \cup Y_1 \subset X_1 \cup Y_1$. The item X_2 is equivalent to a proper subset $Y_2 \cup Y_1$, so it is non-generator. The contrapose gives the result. \square

This theorem is central to the research space, no scanning is done if the candidate is not generator. Only the generator candidates are generated from the database. Fortunately, they are at lower levels. This means that at a certain level, no candidate is generator, no need to browse the database. Its execution process is described by Algorithm 1 below. The algorithm takes as argument a context \mathcal{B} , a minimum support *minsup*. It returns a set \mathcal{F} of frequent patterns, where \mathcal{C}_k denotes the set of candidate k -itemsets and \mathcal{CGM}_k the set of candidate generator k -itemsets. The database \mathcal{B} is built in line 1. Next, \mathcal{F}_1 and \mathcal{F}_2 are generated in a single pass (Algorithm 1 lines 2 and 3). The EOMF-Gen procedure (Algorithm 2) is called to generate candidates (Algorithm 1 line 5). It takes as argument \mathcal{F}_{k-1} , and returns a superset C_k . The initialization of C_k to the empty set is done in line 1 (Algorithm 2). A join between the elements of \mathcal{F}_{k-1} is then made (Algorithm 2 lines 2 to 6). Indeed, two p and q items of \mathcal{F}_{k-1} form a c if, and only if they contain common $(k-2)$ -itemsets. For example, joining ABC and ABD gives $ABCD$. However, joining ABC and CDE does not work because they do not contain common 2-itemsets. Once C_k has been established, it researches among the elements of C_k . If this is the case, it calculates the support in two cases (Algorithm 1 lines 9 to 13): if c is generator, an access to the database is made to know its support (Algorithm 1 line 10), otherwise, it is derived from its subsets without going through the database (Algorithm 1 line 12). The support is then increased (Algorithm 1 line 14). And, only frequent patterns are retained in \mathcal{F}_k

Algorithm 1. Extraction of frequent itemset (EOMF)

Require: A dataset $\mathcal{B} = (\mathcal{T}, \mathcal{I}, \mathcal{R})$, a minimum support *minsup*.

Ensure: A set \mathcal{F} of frequent patterns.

```

1: MATRICESUPPORT  $\leftarrow$  Scan( $\mathcal{B}$ ); //Scan dataset  $\mathcal{B}$ 
2:  $\mathcal{F}_1 \leftarrow \{c_1 \in \text{MATRICESUPPORT} \mid \text{supp}(c_1) \geq \text{minsup}\}$ ; //Generate 1-itemsets
3:  $\mathcal{F}_2 \leftarrow \{c_2 \in \text{MATRICESUPPORT} \mid \text{supp}(c_2) \geq \text{minsup}\}$ ; //Generate 2-itemsets
4: for ( $k = 3; \mathcal{F}_{k-1} \neq \emptyset; k++$ ) do
5:    $C_k \leftarrow$  EOMF-Gen( $\mathcal{F}_{k-1}$ ); //New candidate (see algorithm 2)
6:   for all (transaction  $t \in \mathcal{T}$ ) do
7:      $C_t \leftarrow$  subset( $C_k, t$ ) or  $C_t = \{c \in C_k \mid c \subseteq t\}$  //Select candidate in  $t$ 
8:     for all (candidate  $c \in C_t$ ) do
9:       if ( $c \in \mathcal{CGM}_k$ ) then
10:         $\text{supp}(c) = |\{t \in \mathcal{T} \mid c \subseteq t\}|/|\mathcal{T}|$ ;
11:       else
12:         $\text{supp}(c) = \min\{\text{supp}(c') \mid c' \subset c\}$ ;
13:       end if
14:        $\text{supp}(c)++$ ;
15:     end for
16:   end for
17:    $\mathcal{F}_k \leftarrow \{c \in C_k \mid \text{supp}(c) \geq \text{minsup}\}$ ; //Generate frequent patterns
18: end for
19: return  $\mathcal{F} = \bigcup_k \mathcal{F}_k$ 

```

Algorithm 2. Procedure EOMF-Gen

```

Require: A set  $\mathcal{F}_{k-1}$  of frequent  $(k-1)$ -itemset
Ensure: A set  $C_k$  of candidate  $k$ -itemset
1:  $C_k \leftarrow \emptyset$ 
2: for all itemset  $p \in \mathcal{F}_{k-1}$  do
3:   for all itemset  $q \in \mathcal{F}_{k-1}$  do
4:     if  $(p[1] = q[1], \dots, p[k-2] = q[k-2], p[k-1] < q[k-1])$  then
5:        $c \leftarrow p \cup q(k-1)$ ; //Generate candidate
6:     end if
7:     for all  $((k-1)$ -subset  $s$  of  $c$ ) do
8:       if  $(s \in \mathcal{F}_{k-1})$  then
9:          $C_k \leftarrow C_k \cup \{c\}$ ;
10:      end if
11:    end for
12:  end for
13: end for
14: return  $C_k$ 

```

(Algorithm 1 line 17). Figure 1 shows an example execution of the Algorithm 1, conducted the small database from Table 1, where Gen. designates a minimal generator.

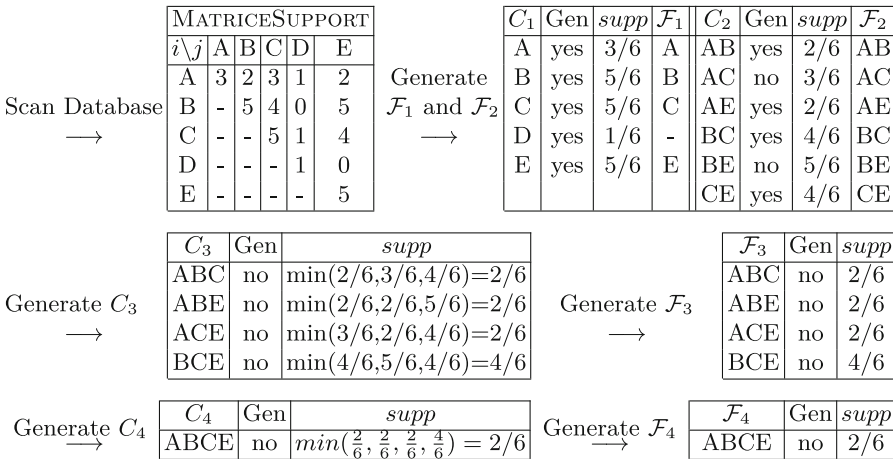


Fig. 1. Example of the Algorithm 1, $minsupp = 2/6$

After reading \mathcal{B} , the itemset D is not frequent. Therefore, it is pruned from the next step. It turns out that $supp(AC) = supp(A)$ and $supp(BE) = supp(B) = supp(E)$, which implies AC and BE are not generators itemsets. Therefore, no candidate of C_3 is generator, it is useless to access the database. Also in the last step, the support of $ABCE$ is equal to that of ABC (or ABE , or ACE).

From this example, our approach does it in a single pass to the database, this is not the case for the existing ones, they do it in 4 passes.

3.2 Extraction of Valid Association Rules

We show that only half of the candidates are to study by using the new technique, *reduction-rules-space*, which divides in two the research space: *attractive zone* and *repulsive zone*. We present our strategies for elimination of uninteresting rules and this reresearch space. First, we are interested in partitioning the space candidate as shown in the following Proposition 2.

Proposition 2. For all $X, Y \in \mathcal{I}$, (1) X fav $Y \Leftrightarrow Y$ fav $X \Leftrightarrow \bar{X}$ fav $\bar{Y} \Leftrightarrow \bar{Y}$ fav \bar{X} . (2) X disfav $Y \Leftrightarrow X$ fav $\bar{Y} \Leftrightarrow \bar{Y}$ fav $X \Leftrightarrow Y$ fav $\bar{X} \Leftrightarrow \bar{X}$ fav Y .

Proof. Let X and Y be items of \mathcal{I} . We first prove, (a) X favors $Y \Leftrightarrow Y$ favors X , (b) X favors $Y \Leftrightarrow \bar{X}$ favors \bar{Y} and (c) X favors $Y \Leftrightarrow \bar{Y}$ favors \bar{X} . In second time, (a) X disfavors $Y \Leftrightarrow X$ favors \bar{Y} , (b) X disfavors $Y \Leftrightarrow \bar{Y}$ favors X , (c) X disfavors $Y \Leftrightarrow Y$ favors \bar{X} and (d) X disfavors $Y \Leftrightarrow \bar{X}$ favors Y .

1(a) X favors $Y \Leftrightarrow P(Y|X) > P(Y) \Leftrightarrow \frac{\text{supp}(X \cup Y)}{\text{supp}(X)} > \text{supp}(Y) \Leftrightarrow \frac{\text{supp}(X \cup Y)}{\text{supp}(Y)} > \text{supp}(X) \Leftrightarrow P(X|Y) > P(X) \Leftrightarrow Y$ favors X . (b) X favors $Y \Leftrightarrow \text{supp}(X \cup Y) > \text{supp}(X)\text{supp}(Y) \Leftrightarrow 1 - \text{supp}(X) - \text{supp}(Y) + \text{supp}(X \vee Y) > 1 - \text{supp}(X) - \text{supp}(Y) + \text{supp}(X)\text{supp}(Y) \Leftrightarrow 1 - \text{supp}(X \wedge Y) > (1 - \text{supp}(X))(1 - \text{supp}(Y)) \Leftrightarrow \text{supp}(\bar{X} \wedge \bar{Y}) > \text{supp}(\bar{X})\text{supp}(\bar{Y}) \Leftrightarrow \frac{\text{supp}(\bar{X} \vee \bar{Y})}{\text{supp}(\bar{X})} > \text{supp}(\bar{Y}) \Leftrightarrow P(\bar{Y}|\bar{X}) > P(\bar{Y}) \Leftrightarrow \bar{X}$ favors \bar{Y} . (c) X favors $Y \Leftrightarrow \text{supp}(\bar{X} \vee \bar{Y}) > \text{supp}(\bar{X})\text{supp}(\bar{Y})$ implies $\frac{\text{supp}(\bar{X} \vee \bar{Y})}{\text{supp}(\bar{Y})} > \text{supp}(\bar{X}) \Leftrightarrow P(\bar{X}|\bar{Y}) > P(\bar{X}) \Leftrightarrow \bar{Y}$ favors \bar{X} .

2(a) X disfavors $Y \Leftrightarrow P(Y|X) < P(Y) \Leftrightarrow 1 - P(Y|X) > 1 - P(Y) \Leftrightarrow P(\bar{Y}|X) > P(\bar{Y}) \Leftrightarrow X$ favors \bar{Y} . (b) X disfavors $Y \Leftrightarrow P(\bar{Y}|X) > P(\bar{Y}) \Leftrightarrow \frac{\text{supp}(X \cup \bar{Y})}{\text{supp}(X)} > \text{supp}(\bar{Y}) \Leftrightarrow P(X|\bar{Y}) > P(X) \Leftrightarrow \bar{Y}$ favors X . (c) X disfavors $Y \Leftrightarrow Y$ disfavors $X \Leftrightarrow P(X|Y) < P(X) \Leftrightarrow P(\bar{X}|Y) > P(\bar{X}) \Leftrightarrow Y$ favors \bar{X} . (d) X disfavors $Y \Leftrightarrow P(\bar{X}|Y) > P(\bar{X}) \Leftrightarrow P(Y|\bar{X}) > P(Y) \Leftrightarrow \bar{X}$ favors Y . \square

This is if X favors Y ($P(Y|X) > P(Y)$), then only $X \rightarrow Y$, $Y \rightarrow X$, $\bar{X} \rightarrow \bar{Y}$ and $\bar{Y} \rightarrow \bar{X}$ are to be studied. If X disfavors Y ($P(Y|X) < P(Y)$), then only $X \rightarrow \bar{Y}$, $\bar{X} \rightarrow Y$, $\bar{Y} \rightarrow X$ and $Y \rightarrow \bar{X}$ are to be studied. That's why our method studies only half of the candidates. We continue the optimization by studying the attractive class rules. To do this, we introduce the Proposition 3 in order to pruning certain positive and négative association rules of the study.

Proposition 3. For all X and Y of \mathcal{I} such that X favors Y and $X \subseteq Y$, we have (1) $M_{GK}(X \rightarrow Y) \leq M_{GK}(Y \rightarrow X)$, (2) $M_{GK}(X \rightarrow Y) = M_{GK}(\bar{Y} \rightarrow \bar{X})$, (3) $M_{GK}(Y \rightarrow X) = M_{GK}(\bar{X} \rightarrow \bar{Y})$, (4) $M_{GK}(X \rightarrow Y) \leq M_{GK}(\bar{X} \rightarrow \bar{Y})$.

Proof. Let X and Y be items of \mathcal{I} . (1) According to Proposition 3(1), X favors $Y \Leftrightarrow Y$ favors X , it gives $M_{GK}(Y \rightarrow X) = \frac{P(X|Y) - P(X)}{1 - P(X)} = \frac{P(X)[P(Y|X) - P(Y)]}{P(\bar{X})P(Y)} = \frac{P(X)P(\bar{Y})}{P(\bar{X})P(Y)} \frac{P(Y|X) - P(Y)}{1 - P(Y)} = \frac{P(X)}{P(\bar{X})} \frac{P(\bar{Y})}{P(Y)} M_{GK}(X \rightarrow Y)$. Because X favors Y and

$X \subseteq Y$, we have $P(X) \geq P(Y) \Leftrightarrow P(\bar{X}) \leq P(\bar{Y}) \Leftrightarrow P(X)P(\bar{Y}) \geq P(\bar{X})P(Y)$, where $M_{GK}(X \rightarrow Y) \leq M_{GK}(Y \rightarrow X)$. (2) $M_{GK}(X \rightarrow Y) = \frac{P(Y|X)-P(Y)}{1-P(Y)} = \frac{-P(X|\bar{Y})+P(X)}{P(\bar{X})} = \frac{P(\bar{X}|\bar{Y})-P(\bar{X})}{1-P(\bar{X})} = M_{GK}(\bar{Y} \rightarrow \bar{X})$. From this property, M_{GK} is implicative. So, the property (3) is immediate, it derives from this implicative character of the M_{GK} . The property remains to be shown (4). Indeed, according to Proposition 3(2), we have $M_{GK}(X \rightarrow Y) \leq M_{GK}(Y \rightarrow X) = M_{GK}(\bar{X} \rightarrow \bar{Y})$, which gives us $M_{GK}(X \rightarrow Y) \leq M_{GK}(\bar{X} \rightarrow \bar{Y})$. \square

In this Proposition 3, the properties (1), (2), (3) and (4) guarantee that if $X \rightarrow Y$ is valid, then $Y \rightarrow X$, $\bar{X} \rightarrow \bar{Y}$ and $\bar{Y} \rightarrow \bar{X}$ will also be the same because M_{GK} of the rule $X \rightarrow Y$ is less than or equal to those of $Y \rightarrow X$, $\bar{X} \rightarrow \bar{Y}$ and $\bar{Y} \rightarrow \bar{X}$. The set of valid rules of the class is thus derived from the only rule $X \rightarrow Y$. This will significantly limit the research space. The following Proposition 4 is introduced to loosen certain rules of the repulsive class.

Proposition 4. *For all X and Y of \mathcal{I} , such that X disfavors Y and $X \subseteq Y$, we have (1) $M_{GK}(X \rightarrow \bar{Y}) = M_{GK}(Y \rightarrow \bar{X})$, (2) $M_{GK}(\bar{X} \rightarrow Y) = M_{GK}(\bar{Y} \rightarrow X)$, and (3) $M_{GK}(X \rightarrow \bar{Y}) \leq M_{GK}(\bar{X} \rightarrow Y)$.*

Proof. Let X and Y of \mathcal{I} . According to Proposition 3(2), we have X disfavors $Y \Leftrightarrow X$ favors $\bar{Y} \Leftrightarrow \bar{Y}$ favors $X \Leftrightarrow Y$ favors $\bar{X} \Leftrightarrow \bar{X}$ favors Y . Thus, due to the implicative character of M_{GK} , the properties (1) and (2) are then immediate. It remains to show (3). As X favors \bar{Y} , we get $M_{GK}(X \rightarrow \bar{Y}) = \frac{P(\bar{Y}|X)-P(\bar{Y})}{1-P(\bar{Y})} = \frac{P(\bar{X})P(\bar{Y})}{P(X)P(\bar{Y})} \frac{P(Y|\bar{X})-P(Y)}{1-P(\bar{Y})} = \frac{P(\bar{X}P(\bar{Y}))}{P(X)P(\bar{Y})} M_{GK}(\bar{X} \rightarrow Y)$. By hypothesis, X disfavors Y and $X \subseteq Y$, we have $P(X) \geq P(Y) \Leftrightarrow P(\bar{X}) \leq P(\bar{Y})$ implice $P(\bar{X})P(\bar{Y}) \leq P(X)P(Y)$, finally $M_{GK}(X \rightarrow \bar{Y}) \leq M_{GK}(\bar{X} \rightarrow Y)$. \square

The properties (1), (2) and (3) of this Proposition 4 indicate that if $X \rightarrow \bar{Y}$ is valid, then $\bar{X} \rightarrow Y$, $Y \rightarrow \bar{X}$ and $\bar{Y} \rightarrow X$ will be valid, because this M_{GK} is less than or equal to those of $\bar{X} \rightarrow Y$, $Y \rightarrow \bar{X}$ and $\bar{Y} \rightarrow X$. Only $X \rightarrow \bar{Y}$ will make it possible to deduce the interest of the class.

Proposition 5. *For all X and Y of \mathcal{I} , $M_{GK}(X \rightarrow \bar{Y}) = -M_{GK}(X \rightarrow Y)$.*

Proof. We show in two cases: (1) X favors Y and (2) X disfavors Y . (1) X favors $Y \Leftrightarrow X$ favors $\bar{\bar{Y}} \Leftrightarrow X$ disfavors \bar{Y} implies $M_{GK}(X \rightarrow \bar{Y}) = \frac{P(\bar{Y}|X)-P(\bar{Y})}{P(\bar{Y})} = -\frac{P(Y|X)-P(Y)}{1-P(\bar{Y})} = -M_{GK}(X \rightarrow Y)$. (2) X disfavors $Y \Leftrightarrow X$ favors \bar{Y} , this gives $M_{GK}(X \rightarrow \bar{Y}) = \frac{P(\bar{Y}|X)-P(\bar{Y})}{1-P(\bar{Y})} = -\frac{P(Y|X)-P(Y)}{P(\bar{Y})} = -M_{GK}(X \rightarrow Y)$. \square

The next result of the following proposition makes it possible to characterize the exact negative association rules according to the implicative measure M_{GK} .

Proposition 6. *Let X, Y, T and Z be four itemsets of \mathcal{I} , such that X favors Y and Z favors T , and $X \cap Y = Z \cap T = \emptyset$, and $X \subset Z \subseteq \gamma(X)$, and $Y \subset T \subseteq \gamma(Y)$. Then, $supp(X \cup Y) = supp(Z \cup T)$ and $M_{GK}(X \rightarrow Y) = M_{GK}(Z \rightarrow T)$.*

Proof. In fact, $\text{supp}(X \cup Y) = \frac{|g(X \cup Y)|}{|\mathcal{I}|} = \frac{|g(X) \cap g(Y)|}{|\mathcal{I}|}$ and $\text{supp}(Z \cup T) = \frac{|g(Z \cup T)|}{|\mathcal{I}|} = \frac{|g(Z) \cap g(T)|}{|\mathcal{I}|}$. Because $X \subset Z \subseteq \gamma(X)$ and $Y \subset T \subseteq \gamma(Y)$, we have $\text{supp}(X) = \text{supp}(Z)$ and $\text{supp}(Y) = \text{supp}(T)$. It causes $g(X) = g(Z)$ and $g(Y) = g(T)$ implies $\text{supp}(X \cup Y) = \text{supp}(Z \cup T)$. As $\text{supp}(X) = \text{supp}(Z)$ and $\text{supp}(Y) = \text{supp}(T)$, we have $P(Y|X) = P(T|Z) \Leftrightarrow P(Y|X) - P(Y) = P(T|Z) - P(Y) \Leftrightarrow \frac{P(Y|X) - P(Y)}{1 - P(Y)} = \frac{P(T|Z) - P(T)}{1 - P(T)} \Leftrightarrow M_{GK}(X \rightarrow Y) = M_{GK}(Z \rightarrow T)$. \square

Proposition 7. *Let $X, Y, T, Z \subseteq \mathcal{I}$, such that X disfavors Y and Z disfavors T , and $X \cap Y = Z \cap T = \emptyset$, and $X \subset Z \subseteq \gamma(X)$, and $Y \subset T \subseteq \gamma(Y)$. Then, $\text{supp}(X \cup Y) = \text{supp}(Z \cup T)$ and $M_{GK}(X \rightarrow \bar{Y}) = M_{GK}(Z \rightarrow \bar{T})$.*

Proof. In fact, $\text{supp}(X \cup Y) = \frac{|g(X \cup Y)|}{|\mathcal{I}|}$ and $\text{supp}(Z \cup T) = \frac{|g(Z \cup T)|}{|\mathcal{I}|}$. Because $X \subset Z \subseteq \gamma(X)$ and $Y \subset T \subseteq \gamma(Y)$, we have $\text{supp}(X) = \text{supp}(Z)$ and $\text{supp}(Y) = \text{supp}(T)$. It causes $g(X) = g(Z)$ and $g(Y) = g(T)$ implies $\text{supp}(X \cup Y) = \text{supp}(Z \cup T)$. Because $\text{supp}(X) = \text{supp}(Z)$ and $\text{supp}(Y) = \text{supp}(T)$, we have $P(Y|X) = P(T|Z) \Leftrightarrow P(\bar{Y}|X) = P(\bar{T}|Z) \Leftrightarrow P(\bar{Y}|X) - P(\bar{Y}) = P(\bar{T}|Z) - P(\bar{T}) \Leftrightarrow \frac{P(\bar{Y}|X) - P(\bar{Y})}{1 - P(\bar{Y})} = \frac{P(\bar{T}|Z) - P(\bar{T})}{1 - P(\bar{T})} \Leftrightarrow M_{GK}(X \rightarrow \bar{Y}) = M_{GK}(Z \rightarrow \bar{T})$. \square

Proposition 8. *Let X, Y be two itemsets, such that X disfavors Y , $\text{supp}(X) \neq 0$ and $\text{supp}(Y) \neq 0$. $M_{GK}(X \rightarrow \bar{Y}) = 1 \Leftrightarrow \text{supp}(X \cup Y) = 0$.*

Proof. Because X disfavors Y , we have X favors \bar{Y} (Proposition 2(a)), which leads to $M_{GK}(X \rightarrow \bar{Y}) = 1 \Leftrightarrow \frac{P(\bar{Y}|X) - P(\bar{Y})}{1 - P(\bar{Y})} = 1 \Leftrightarrow P(\bar{Y}|X) - P(\bar{Y}) = 1 - P(\bar{Y}) \Leftrightarrow P(\bar{Y}|X) = 1 \Leftrightarrow P(Y|X) = 0 \Leftrightarrow \frac{\text{supp}(X \cup Y)}{\text{supp}(X)} = 0 \Leftrightarrow \text{supp}(X \cup Y) = 0$. \square

The next Corollaries 1 and 2 are the consequence of the above Proposition 8.

Corollary 1. *For all $X, Y, Z \in \mathcal{I}$, $M_{GK}(X \rightarrow \bar{Y}) = 1$ and $\text{supp}(Y \cup Z) > 0$ implies $M_{GK}(X \rightarrow \overline{Y \cup Z}) = 1$.*

Proof. According to Proposition 8, $\text{supp}(X \rightarrow Y) = 0$. So, for all Z , we have $\text{supp}(X \cup (Y \cup Z)) = 0$. On the other hand, if Z such that $\text{supp}(Z \cup Y) > 0$, then we obtain, again by Proposition 8, that $M_{GK}(X \rightarrow \overline{Y \cup Z}) = 1$. \square

Corollary 2. *For all $X, Y, Z \in \mathcal{I}$, such that $Z \subset X$, $M_{GK}(X \rightarrow \bar{Y}) = 1$ and $\text{supp}(Z \cup Y) = 0$ implies $M_{GK}(Z \rightarrow \bar{Y}) = 1$.*

Proof. For all an itemset Z , such that $Z \subset X$, we have $\text{supp}(X) > 0$. So, if $\text{supp}(Z \rightarrow Y) = 0$, then by Proposition 8, we have $M_{GK}(Z \rightarrow \bar{Y}) = 1$. \square

Proposition 9. *Let $X, Y, Z \subset \mathcal{F}$. If $X \rightarrow Y \setminus X$ is non-valid, then $\tilde{X} \rightarrow Y \setminus \tilde{X}$ is also for any $\tilde{X} \subset X$. It follows that for a rule $Y \setminus Z \rightarrow Z$ to hold, all rules of the form $Y \setminus \tilde{Z} \rightarrow \tilde{Z}$ must also hold, where \tilde{Z} is a non-empty subset of Z .*

This proposition is very ideal, it allows not to consider all possible sets of \mathcal{F} . It indicates that, for a given large itemset, if a rule with consequent c holds then so do rules with consequents that are subsets of c . For example, if the rule

$AB \rightarrow CD$ holds, then the rules $ABC \rightarrow D$ and $ABD \rightarrow C$ must also hold. Conversely, if $ABC \rightarrow D$ is not valid, then $AB \rightarrow CD$ is also not valid.

These different optimizations will be synthesized by the following Algorithm 3. It is initialized by the empty set in line 1. Next, for each itemset of \mathcal{F} set, the set \mathcal{A} is generated (line 3). For each subset X_{k-1} of \mathcal{A} (line 4), the algorithm proceeds in two recursive steps. The first consists in generating attractive class rules using the single rule $X \rightarrow Y$ (Algorithm 3 lines 6 to 11). Indeed, if $supp(X \rightarrow Y) \geq minsup$ and $M_{GK}(X \rightarrow Y) \geq minmgk$, then the \mathcal{E}_{RAPN} set is updated by adding $X \rightarrow Y, Y \rightarrow X, \bar{Y} \rightarrow \bar{X}$ and $\bar{X} \rightarrow \bar{Y}$ (Algorithm 3 line 9). The second step consists in generating repulsive class rules by studying only $X \rightarrow \bar{Y}$ (Algorithm 3 lines 11 to 16). The Algorithm 3 is updated by adding $X \rightarrow \bar{Y}, Y \rightarrow \bar{X}, \bar{Y} \rightarrow \bar{X}$ and $\bar{X} \rightarrow Y$ (Algorithm 3 line 14). ERAPN returns the \mathcal{E}_{RAPN} set (Algorithm 3 line 19).

Algorithm 3. Extraction of valid association rules

Require: A set \mathcal{F} of frequent patterns, a *minsup* and *minmgk*.

Ensure: A set \mathcal{E}_{RAPN} of valid positive and negative rules.

```

1:  $\mathcal{E}_{RAPN} = \emptyset$ ;
2: for all ( $k$ -itemset  $X_k$  of  $\mathcal{F}$ ,  $k \geq 2$ ) do
3:    $\mathcal{A} = \{(k-1)\text{-itemset} \mid X_{k-1} \subset X_k\}$ 
4:   for all ( $X_{k-1} \in \mathcal{A}$ ) do
5:      $X = X_{k-1}; Y = X_k \setminus X_{k-1}$ ;
6:     if ( $P(Y|X) > P(Y)$ ) then
7:        $supp(X \cup Y) = \frac{|g(X \cup Y)|}{|\mathcal{T}|}$ ;  $M_{GK}(X \rightarrow Y) = \frac{P(Y|X) - P(Y)}{1 - P(Y)}$ ;
8:       if ( $supp(X \cup Y) \geq minsup$  &  $M_{GK}(X \rightarrow Y) \geq minmgk$ ) then
9:          $\mathcal{E}_{RAPN} \leftarrow \mathcal{E}_{RAPN} \cup \{X \rightarrow Y, Y \rightarrow X, \bar{Y} \rightarrow \bar{X}, \bar{X} \rightarrow \bar{Y}\}$ ;
10:      end if
11:     else
12:        $supp(X \cup \bar{Y}) = \frac{|g(X \cup \bar{Y})|}{|\mathcal{T}|}$ ;  $M_{GK}(X \rightarrow \bar{Y}) = \frac{P(\bar{Y}|X) - P(\bar{Y})}{1 - P(\bar{Y})}$ ;
13:       if ( $supp(X \cup \bar{Y}) \geq minsup$  &  $M_{GK}(X \rightarrow \bar{Y}) \geq minmgk$ ) then
14:          $\mathcal{E}_{RAPN} \leftarrow \mathcal{E}_{RAPN} \cup \{X \rightarrow \bar{Y}, Y \rightarrow \bar{X}, \bar{Y} \rightarrow \bar{X}, \bar{X} \rightarrow Y\}$ ;
15:       end if
16:     end if
17:   end for
18: end for
19: return  $\mathcal{E}_{RAPN}$ 

```

Example Illustrate of Algorithm 3. We consider the frequent itemset ABC of \mathcal{F} (cf. Fig. 1), $minsup = 0.1$ and $minmgk = 0.6$. Will give us the candidates: $BC \rightarrow A, AC \rightarrow B, AB \rightarrow C, A \rightarrow BC, B \rightarrow AC$ and $C \rightarrow AB$. Because ABC is frequent, then A, B and C are also frequent, which gives $A \rightarrow B, A \rightarrow C$ and $B \rightarrow C$. To do this, we generate the one-item conclusion rules. We combine those which are valid in order to obtain the two-item conclusion.

Indeed, we obtain $M_{GK}(A \rightarrow B) = -1$, then $A \rightarrow B$ is not interesting. This leaves us to study the derived negative rules. According to Proposition 5, $M_{GK}(A \rightarrow \bar{B}) = -M_{GK}(A \rightarrow B) = 1 > 0.6$. Moreover, $supp(A\bar{B}) = 0.33 > 0.1$, then $A \rightarrow \bar{B}$ is valid. Because $supp(BC) = 0.67 > 0$, we have $M_{GK}(A \rightarrow \overline{BC}) = 1$ and $supp(A\overline{BC}) = -0.17$. So, $A \rightarrow \overline{BC}$ is invalid. In other words, because $A \rightarrow B$ is invalid, its derived $A \rightarrow BC$ is also invalid. But $supp(ABC) = 0.33 > 0.1$ and $M_{GK}(AB \rightarrow C) = 1 > 0.6$, $AB \rightarrow C$ is valid.

Because $M_{GK}(B \rightarrow C) = -0.21 < 0$, $B \rightarrow C$ is invalid, $B \rightarrow AC$ is also invalid. But, $supp(\overline{BC}) = supp(C) - supp(BC) = 0.17 > 0.1$ and $M_{GK}(\overline{B} \rightarrow C) = 1 > 0.6$, $\overline{B} \rightarrow C$ is valid. Because $supp(\overline{BAC}) = supp(AC) - supp(B) = 1 > 0.1$ and $M_{GK}(\overline{B} \rightarrow AC) = 1 > 0.6$, $\overline{B} \rightarrow AC$ is also valid.

Because, $M_{GK}(A \rightarrow BC) = 0$, and $M_{GK}(BC \rightarrow A) = -0.5 < 0$, and $M_{GK}(BC \rightarrow \overline{A}) = 0.5 < 0.6$, then the rules $A \rightarrow BC$, $BC \rightarrow A$, and $BC \rightarrow \overline{A}$ are invalid. But $supp(AC) = 0.5 > 0.1$ and $M_{GK}(A \rightarrow C) = 1$, $A \rightarrow C$ is valid. Next, because $M_{GK}(C \rightarrow AB) = 0.1 < 0.6$ and $M_{GK}(AC \rightarrow \overline{B}) = -M_{GK}(AC \rightarrow B) = 1 > 0.6$, but $supp(AC\overline{B}) = -0.33 < 0$, the rules $C \rightarrow AB$, $AC \rightarrow B$ and $AC \rightarrow \overline{B}$ are invalid. The Table 4 summarizes these results.

Table 4. Rules extracted on the itemset ABC of \mathcal{F}

\mathcal{E}_{RAPN}	Support	Confidence	M_{GK}
$A \rightarrow C$	0.5	1	1
$A \rightarrow \overline{B}$	0.33	1	1
$\overline{B} \rightarrow C$	0.17	1	1
$AB \rightarrow C$	0.33	1	1
$\overline{B} \rightarrow AC$	1	1	1

Our approach restores five rules in total, including two positive rules of type $X \rightarrow Y$, two negative rules of type $\overline{X} \rightarrow Y$, and a negative rule of type $X \rightarrow \overline{Y}$.

4 Experimental Results

This section presents the experimental study conducted in order to evaluate the performances of our algorithm. The latter is implemented in R and tested on PC Core i3 and 4 GB of RAM running under Windows system. We compare the results with those of Wu and RAPN, conducted out on four databases from UCI, such as **Adult**, **German**, **Income** and **Iris**. For each algorithm, we have chosen the same thresholds to avoid biasing the results. The following Table 5 reports the characteristics of datasets, and the number of positive and negative rules by varying the minimum thresholds $minsup$ and $minmgk$. Indeed, the first three columns indicate the data characteristics in question, the last fifteen columns present the different results, where the column labelled “++” corresponds to the type $X \rightarrow Y$, column “±” to $\overline{X} \rightarrow Y$, column “±” to $X \rightarrow \overline{Y}$, and “--” to $X \rightarrow \overline{Y}$. The behaviour of algorithms varies according to data characteristics. The large database is much more time-consuming to run. In other words, the number of rules increases as thresholds decrease. Except for dense databases (**Adult** and **German**) and relatively low thresholds ($minsup = 1\%$ et $minmgk = 60\%$), the number of rules (see Table 5) in Wu is 100581 and 89378 for RAPN. They are relatively large, due to the strong contribution of positive rules of type $X \rightarrow Y$ and negative rules of type $X \rightarrow \overline{Y}$ (see Table 5), than for ERAPN

Table 5. Characteristic of Datasets and Results extracts

Database	\mathcal{T}	\mathcal{I}	<i>minsup</i>	<i>minmgk</i>	Wu				RAPN				ERAPN						
					++	±	±	--	∑	++	±	±	--	∑	++	±	±	--	∑
Adult	48842	115	1%	60%	97956	625	1215	785	100581	87800	542	615	421	89378	27500	422	510	352	28784
			2%	70%	55925	453	852	556	57786	53950	323	503	344	55120	25536	354	385	225	26500
			3%	80%	38750	345	412	310	39817	22033	156	254	145	22588	18523	124	154	124	18925
German	1000	71	1%	60%	51478	456	1148	555	53637	41235	401	565	412	42613	26456	340	380	245	27421
			2%	70%	39683	352	744	456	41235	38555	234	425	384	39598	18800	220	203	156	19379
			3%	80%	9835	144	545	321	10845	18500	75	325	232	19132	12157	95	85	55	15392
Income	6876	50	1%	60%	2800	227	527	254	3808	2130	350	385	286	3151	1552	95	103	84	1834
			2%	70%	2200	127	327	213	2867	2054	214	330	185	2783	1433	55	63	65	1616
			3%	80%	1325	87	252	121	1785	1212	65	156	45	1478	923	35	30	17	1005
Iris	150	15	1%	60%	2437	159	196	160	2952	1954	10	60	24	2048	1500	150	165	124	1939
			2%	70%	2000	159	145	120	2424	1323	10	55	20	1407	1122	75	85	65	1347
			3%	80%	1200	159	59	45	1463	1056	25	30	-	1111	965	14	22	18	1019

(28784 rules). The rules of type $\bar{X} \rightarrow Y$ and $\bar{X} \rightarrow \bar{Y}$ remain reasonable for each algorithm. On less dense databases (*Income* and *Iris*), the 3 algorithms give reasonable numbers. Note that RAPN, for *Iris* data, does not extract the type $\bar{X} \rightarrow \bar{Y}$ (see Table 5) for *minsup* (resp. *minmgk*) over 3% (resp. 80%). Figure 2 below shows the response times by varying the *minupp* and keeping *minmgk* = 60%.

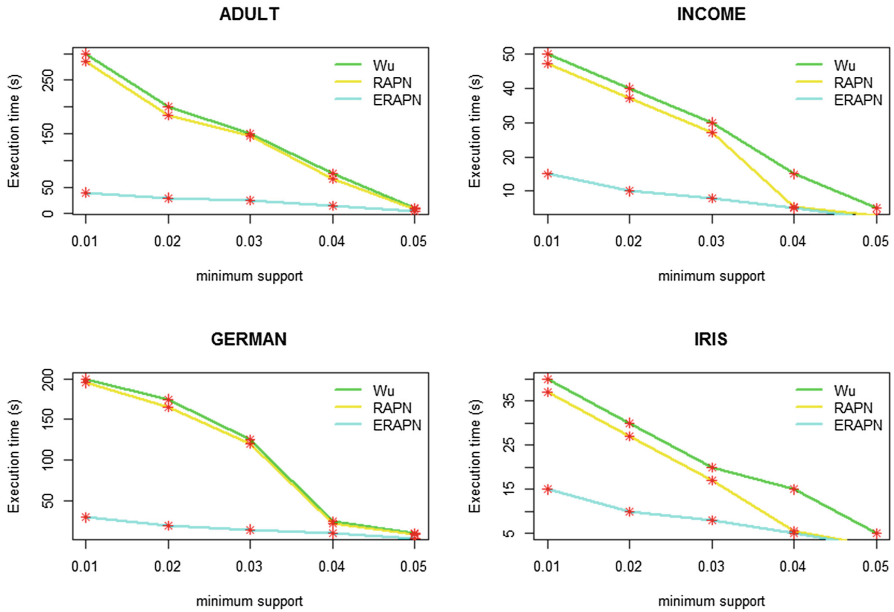


Fig. 2. Performances for each algorithm according to *minsup*

They also increase when thresholds are lowered. The execution time of ERAPN is faster than that of Wu and RAPN. ERAPN gained 7 more times the best response speed in the worst cases. These different performances can be explained as follows. RAPN and Wu are limited to the classic data structure, which requires repetitive access to the database. The Wu has the lowest performance. One of the main reasons lies in the pruning technique. The *interest* measure used does not have effective properties for trimming the space of frequent patterns. In addition, no optimized technique is used, the space of candidates can be covered exhaustively. Our algorithm introduces different optimizations for the research space. Therefore, the space of frequent patterns can be traversed only once. Moreover, the space of candidate rules is only half full. In all cases, the ERAPN algorithm is excellent to the execution time and the quality of valid association rules. So, it is the most selective and concise for knowledge extraction.

5 Conclusion

In this paper, we have studied the problem of positive and negative association rules for big data. Further optimizations have been defined. Experiments conducted on reference databases, compared to RAPN and Wu algorithms, have emphasized the efficiency of our algorithm. A study on the extraction of disjunctions/conjunctions association rules has not been initially developed, which gives leads to explore from a methodological and algorithmic point of view.

References

1. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In Proceedings of ACM SIGMOD, pp. 207–216 (1993)
2. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proceedings of 20th VLDB Conference, Santiago, Chile, pp. 487–499 (1994)
3. Bemarisika, P., Totohasina, A.: EOMF, un algorithme d'extraction optimisée des motifs fréquents. In: Proceedings of AAFD & SFC, Marrakech, Maroc, pp. 198–203 (2016)
4. Bemarisika, P.: Extraction de règles d'association selon le couple support- M_{GK} : Graphes implicatifs et Application en didactique des mathématiques. Université d'Antananarivo, Madagascar (2016)
5. Bemarisika, P., Totohasina, A.: Optimized mining of potential positive and negative association rules. In: Bellatreche, L., Chakravarthy, S. (eds.) DaWaK 2017. LNCS, vol. 10440, pp. 424–432. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-64283-3_31
6. Brin, S., Motwani, R., Silverstein, C.: Beyond market baskets: Generalizing association rules to correlation. In: Proceedings of the ACM SIGMOD, pp. 265–276 (1997)
7. Cornelis, C., Yan, P., Zhang, X., Chen, G.: Mining positive and negative association rules from large databases. In: Proceedings of the IEEE, pp. 613–618 (2006)
8. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Heidelberg (1999). <https://doi.org/10.1007/978-3-642-59830-2>
9. Guillaume, S.: Traitement de données volumineuses: Mesure et algorithmes d'extraction de règles d'association. Ph.D. thesis, Université de Nantes (2000)
10. Guillaume, S., Papon, P.-A.: Extraction optimisée de règles d'association positives et négatives (RAPN). In: Actes de la 13e Conf. Int., pp. 157–168. EGC, Franco (2013)
11. Savasere, A., Omiecinski, E., Navathe, S.: Mining for strong negative associations in a large database of customer transactions. In: Proceedings of ICDE, pp. 494–502 (1998)
12. Teng, W.-G., Ming-Jyh, H., Ming-Syan, C.: A statistical framework for mining substitution rules. Knowl. Inf. Syst. **7**, 158–178 (2005)
13. Totohasina, A., Ralambondrainy H.: ION, a pertinent new measure for mining information from many types of data. In: IEEE, SITIS, pp. 202–207 (2005)
14. Hämäläinen, W.: Kingfisher: an efficient algorithm for searching for both positive and negative dependency rules with statistical significance measures. Knowl. Inf. Syst. **32**, 383–414 (2012)
15. Wu, X., Zhang, C., Zhang, S.: Efficient mining of both positive and negative association rules. ACM Trans. Inf. Syst. **3**, 381–405 (2004)



Multistep-ahead Prediction: A Comparison of Analytical and Algorithmic Approaches

Fouad Bahrpeyma^(✉), Mark Roantree, and Andrew McCarren

Insight Centre for Data Analytics, School of Computing,
Dublin City University, Dublin 9, Ireland

Fouad.Bahrpeyma2@mail.dcu.ie, {mark.roantree, andrew.mccarren}@dcu.ie

Abstract. Most approaches to forecasting time series data employ one-step-ahead prediction approaches. However, recently there has been focus on multi-step-ahead prediction approaches. These approaches demonstrate enhanced prediction capabilities. However, multi-step-ahead prediction increases the complexity of the prediction process in comparison to one-step-ahead approaches. Typically, studies in the examination of multi-step ahead methods have addressed issues such as the increased complexity, inaccuracy, uncertainty, and error variance on the prediction horizon, and have been deployed in various domains such as finance, economics, agriculture and hydrology. When determining which algorithm to use in a time series analyses, the approach is to analyze the series for numerous characteristics and features, such as heteroscedasticity, auto-correlation, seasonality and stationarity. In this work, a comparative analysis of 20 different time series datasets is presented and a demonstration of the complexity in deciding which approach to use is given. The study investigates some of the main prediction approaches such as ARIMA (Autoregressive integrated moving average), NN (Neural Network), RNN (Recurrent neural network) and SVR (Support vector regression), which focus on the recursive prediction strategy and compare them to a new approach known as MRFA (Multi-Resolution Forecast Aggregation).

1 Introduction

A time series data-set can typically be described as a series of values or quantities that are ordered in a time sequenced fashion, often with equal intervals between them [4]. In the past time series analysis was typically, but not exclusively applied to traditional econometric problems [15]. However, with the advent of Big data sources such as sensor technology, temporal and streaming data, one can certainly consider these data sources as time series problems. Generally, when predicting forward using such data, one can either implement one-step

This work is supported by Science Foundation Ireland under grant number SFI/12/RC/2289.

ahead prediction (OSAP) or multi-step-ahead prediction (MSAP) approaches [3]. Both approaches have advantages and disadvantages when making predictions. OSAP methods typically suffer from expanding error variance when going beyond the short term prediction horizon, while MSAP methods can be difficult to construct, as one is using complex RNN (Recurrent neural network) or LSTM (Long short term memory) algorithms, and determining an appropriate number of hidden nodes and layers is less than trivial. Additionally, each time series has hidden attributes such as heteroscedasticity, auto-correlation, seasonality and stationarity. These attributes can be measured using various metrics that have been constructed over the years [5,20]. Certain fields of interest such as the Agri or Finance sectors generate large amounts of univariate time series data. Traditionally, one would consider these metrics when applying analytical methods such as ARIMA or ARCH (Auto regressive conditional heteroscedasticity). However, when choosing modern machine learning applications in time series prediction problems these characteristics are rarely considered.

Motivation. When determining which approach to use in a time series prediction problem, one is often faced with the significant task of selecting one method/algorithm from a large selection of algorithms that can be derived for example, from the ARIMA, SVR, NN, RNN and LSTM families. By creating a list of defined criteria that characterize a time series one can compare a new time series with work previously carried out using the meta characteristics that were generated in the historical analysis. This then allows researchers to choose an algorithm that predicts more accurately in the required prediction horizon, and with minimum computational and research effort.

Contribution. In this paper, an extensive set of time series metrics and a wide portfolio of OSAP and MSAP methods on disparate data streams and sources are implemented to guide the analyst in choosing the most appropriate algorithm. A detailed analysis and classification of each approach is given, which in turn, provides a meta analysis for each series.

Paper Structure. The paper is structured as follows: in Sect. 2, we discuss related research in time series analysis; In Sect. 3, a description of the time series characteristic metrics are outlined; Sect. 4 describes the classification evaluation procedure and provides an analysis of the disparate data sources; and finally, in Sect. 5, we present our conclusions.

2 Related Research

Various measures are implemented in time series analytical approaches [8]. Assessing the characteristics of a time series have prevalently focused on measuring the series for auto-correlation, seasonality, hetroscedasticity, non-linearity, volatility and non-stationarity [15]. In this paper we intend to extend the analyses with a few more measures to conduct a study on their presence with respect to the performance of time series prediction.

Time series prediction has long been practiced using the ARIMA family. The ARIMA family fundamentally relies on two main components, an Autoregressive (AR) model and a Moving Average (MA) model [4]. From the machine learning point of view, time series prediction problems are fundamentally sequential supervised learning problems, and need to be converted to classical supervised learning, in order to solve them using machine learning techniques. NNs are the most popular ML technique used for prediction purposes and were inspired from the human brain's neural inference system [6]. RNNs by incorporating feedback connections extend the memory of NNs with improved performance in particular for time series prediction [17]. LSTM is an extension of RNNs with an enhanced ability in capturing long term memory from the given time series [12]. SVR is the extension of the well known SVM classifiers used for regression purposes [9].

Multi-step ahead prediction has predominantly been implemented using the principles of either the direct or the recursive strategies. The main disadvantage of the direct strategy is that serial correlations are discarded while the recursive strategy suffers from accumulation of errors. The recursive strategy has generally been seen as the more popular method as it attempts to use the inherent memory of a time series by using its natural serial correlation. Thus, in this research we focus on the recursive strategy. In the recursive strategy [18], an OSAP model (which can be a machine learning regression model) is run multiple times over the given prediction horizon. For a signal y , the output of the OSAP model $f(\cdot)$ is calculated by Eq. 1:

$$y_{t+1} = f(y_t, \dots, y_{t-d+1}) \quad (1)$$

Where d is the number of input lags. In Eq. 1, $f(\cdot)$ can be modeled using NN, RNN, LSTM or SVR as the OSAP method to implement the recursive strategy. There is also another method called Multi-Resolution Forecast Aggregation (MRFA) [2] that follows the principles of the recursive strategy to implement MSAP but not exactly through Eq. 1. MRFA incorporates a concept known as the Resolution of Impact (ROI) and analyses the impact of local patterns on the future of the signal; resolution refers to the length of the horizon for which this impact is studied. Using MRFA, the value of the signal in the desired prediction horizon is predicted at multiple resolutions using separate forecast models. The predicted values are then aggregated to gain the final forecast.

3 Time Series Characteristics and Metrics

In traditional econometric modelling, an analysis to determine the characteristics of the time series is undertaken. This will generally help the analyst in determining which algorithm or analytical tool to use. However, most studies only rely on a few basic characteristics such as stationarity and auto-correlation, which cannot sufficiently describe time series properties. In this section, some prediction metrics that help determine the most appropriate algorithm are presented in an attempt to provide a more informative analysis.

3.1 Gaussianity and Sampling Frequency

Gaussianity is a measure of how similar the time series (or the fluctuations/error) is to a normal distribution, and can be measured via the D'Agostino's K-squared test [10]. The frequency at which the system is sampled contributes to the complexity of the time series. Typically, the complexity of a low frequency signal is higher than that of high frequency one. In a low frequency signal, each sample is predominantly an aggregation over a period of time. This aggregation, in particular, tends to increase the Gaussianity of the data, which consequently enhances the complexity of the modeling process [7]. In contrast, in high frequencies, noise or temporal anomalies appear more salient in the formation of the signal. Various methods will show differing performances when dealing with such noise effects [5].

3.2 Trend and Seasonality

Trend is a pattern in a time series which can be caused by high level factors such as socio-economic or political forces. Depending on the data, the contribution of trend to a time series can be additive or multiplicative [4]. Seasonality occurs as regular patterns or fluctuations in a time series, [5], and can be caused by the influence of factors such as season, month, or day of the week [4]. After eliminating trend and seasonality in the time series, the remaining components are usually referred to as white noise or irregular component [8].

Like trend, seasonality may be additive or multiplicative:

$$\begin{cases} I_t + T_t + S_t \rightarrow \text{additive}, \\ I_t \times T_t \times S_t \rightarrow \text{Multiplicative}. \end{cases} \quad (2)$$

where y_t, I_t, T_t , and S_t are the original time series, irregular component, trend, and seasonality, respectively. One approach to detect seasonality is to analyze sample auto-correlation function (ACF) and partial auto-correlation function plots (PACF) [1].

3.3 ACF and PACF Plots

Sample auto-correlation function (ACF) and partial auto-correlation function (PACF) help provide metrics in assessing the level of auto-correlation, seasonality and to a lesser extent the levels of non-stationarity. The ACF plot represents lagged correlations in a time series in terms of correlation coefficients. For the time series y of length T , the ACF is computed as [1]:

$$r_T = \frac{c_\tau}{c_0}, \tau = 0, \dots, T - 1 \quad (3)$$

where c_0 is the sample variance, and is the empirical auto-covariance at lag τ :

$$c_\tau = \frac{1}{T} \sum_{t=\tau}^{T-1} (y_t - \bar{y})(y_{t-\tau} - \bar{y}) \quad (4)$$

The PACF plot illustrates the partial correlation coefficients between the given time series and its lags. PACF is the estimated lag- h coefficient in an AR model containing h lags adjusting for the lags between time point 1 and lag h [1], i.e.:

$$\begin{cases} \phi_{1,1}(y) = \text{corr}(y_{t+1}, y_t) = \rho(1) \\ \phi_{h,h}(y) = \text{corr}(y_{t+h} - \hat{y}_{t+h}, y_t - \hat{y}_t); h \geq 2 \end{cases} \quad (5)$$

where $\text{corr}(x,y)$ is the correlation function, defined as:

$$\text{corr}(x, y) = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n \sum x_i^2 - (\sum x_i)^2} \sqrt{n \sum y_i^2 - (\sum y_i)^2}} \quad (6)$$

3.4 Stationarity

Stationarity is a favourable feature in time series data and can be described as having a steady mean, variance and auto-correlation over the time span of the series [1]. As a result, some mathematical transformations have been used to transform time series into a stationary data-set. Differencing is the most common approach used to reduce the effects of non-stationarity, [4]. However, real world time series tend to exhibit non-stationarity even after differencing. In such cases, other data pre-processing techniques such as de-trending or de-seasonalizing may be required, [8]. Non-stationarity can be identified using unit-root tests. Some popular unit-root tests are the Augmented Dickey-Fuller (ADF) test [11] and the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test [16].

3.5 Conditional Heteroskedasticity

Heteroscedasticity refers to the presence of non-constant variance in a time series over time, which can be conditional or unconditional. The phenomenon is challenging in the unconditional form, where the time series is characterized by non-constant volatility such that future periods of high and low volatility cannot be identified [1]. The Autoregressive conditional heteroscedasticity (ARCH) family can be used to model such phenomena when conditional heteroscedasticity (CH) exists. CH can also be evaluated by applying Ljung-Box Q test to the squared residual series [15].

3.6 Long Range Dependence and Sample Entropy

Long-range dependence (LRD) or long term memory is a feature that is associated with the predictability of a time series. LRD measures the statistical dependence between two differing time points. LRD is present if the statistical dependence declines at a slower than an exponential rate of decay. The Hurst exponent [13] and the Detrended Fluctuation Analysis (DFA) coefficients [14], for example, are two metrics that evaluate a time series for long range dependence. Sample entropy is a metric to assess the complexity of time-series data. For a signal y with sample size N and tolerance r , sample entropy is the negative logarithm of the conditional probability that a sub-series of length m matches point-wise with the next point with tolerance (distance less than) r [19].

4 Evaluation

20 different time series were chosen for analysis in this paper and were taken from disparate monthly, weekly, daily and hourly data sources. The monthly series were lake Erie levels (1921–1970), monthly milk production pounds or `m_m_p` (1962–1974), and the number of employed persons in Australia (1978–1991). The weekly time series were two Irish beef prices (2011–2018), Irish pig prices, 2007–2016, German pig prices (2008–2016), Canadian barley (2008–2016), and German feed barley (2008–2016). The daily time series are foreign exchange rates (1979–1998), minimum temperatures in Melbourne (1981–1990), total female births in California (1959–1959), and mean temperature of the fisher River (1988–1991), bike share variables (2011–2012) and births in USA between 1978 and 2015. Hourly data was taken from one series measuring hourly carbon dioxide emissions.

In order to characterize each dataset, differing tests and metrics were applied. ACF and PACF plots were analyzed to confirm the existence of seasonality. CH was examined using the Ljung-Box Q test on the squared residual series. Also, the Hurst exponent and DFA were conducted on the data to characterize non-stationarity in the series. Table 1 shows these characteristics as well as the frequency, sample size and type/status of non-stationarity for each of the original time series. Gaussianity is evaluated using D’Agostino’s K^2 test and denoted by H for high, L for low, V for very low, and 0 for no Gaussianity.

In Table 1 which shows the results for 8-steps-ahead prediction, `m_l_e_l`, `m_m_p`, and `m_n_o_e` indicate lake Erie water levels, monthly milk production pounds, and the level of employment in Australia, respectively (Monthly data). Also, `H_Um_3P` and `S_Um_2P` indicate two Irish beef prices, `IreCent` is the Irish pig price, `GerCent` is the German pig prices, `CanBar` is Canadian barley prices and `GerFBar` is the German feed barley prices (weekly data). Daily foreign exchange rates are denoted by `d_f_ex_r`, daily minimum temperatures in Melbourne by `d_m_t`, daily total female births in California by `d_t_f_b`, mean daily temperature of fisher River near Dallas by `m_d_t_f`, daily bike share variables by `d_b_sh1`, `d_b_sh2` and `d_b_sh4`, a US economic series by `Ec_unem`, births in US in 1978 and 2015 by `US_B_78` and `US_B_15`, (Daily data) and hourly carbon dioxide emission by `LNOxEm`.

4.1 Results

In Table. 1, for each series the method with the minimum prediction error is reported as the preferred method. The prediction error is calculated as the average normalised mean square error. For each series the order of the $SARIMA(p, d, q)$ model was obtained using the exact maximum likelihood using a Kalman filter. A detailed discussion on optimal models is beyond the scope of this paper. However, details on model parameters selection can be found in [21, 22]. The experimental results based the application of the methods discussed are outlined in Table 1.

Table 1. Experimental results

Frequency	Time series	Gaussianity	Sample size	Seasonality	Stationarity	CH	Hurst exponent	DFA	Sample entropy	Type	Min error
M	m_le1	L	600	12	N	Y	0.59	1.17	0.88	Non-stationary	MRFA
M	m_m_p	L	156	12	N	N	0.69	1.52	0.68	Brownian noise	MRFA
M	m_n_o_e	V	759	12	N	Y	0.97	1.28	0.29	Non-stationary	NN
W	H_Um_3P	H	357	52	N	Y	0.93	1.79	0.41	Non-stationary	MRFA
W	S_Um_2P	L	357	52	N	Y	0.92	1.68	0.60	Non-stationary	SVR
W	IreCent	V	512	52	N	Y	0.92	1.86	0.32	Non-stationary	MRFA
W	GerCent	V	468	52	N	Y	0.86	1.58	0.53	Non-stationary	NN
W	CanBar	V	468	52	N	N	0.88	1.64	0.20	Non-stationary	LSTM
W	GerFBar	V	468	52	N	N	0.86	1.63	0.25	Non-stationary	NN
D	d_f.ex_r	V	4774	365	N	Y	0.95	1.52	0.05	Brownian noise	SVR
D	d_m.t	V	3650	365	Y	N	0.90	1.08	1.62	Pink noise	ARIMA
D	d.t.f.b	V	365	-	Y	Y	0.61	0.69	2.20	Stationary	NN
D	m_d.t.f	V	1461	365	N	N	0.88	1.24	0.73	Non-stationary	MRFA
D	d_b.sh4	V	731	-	N	Y	0.51	0.66	2.05	White noise	ARIMA
D	d_b.sh2	V	731	-	N	N	0.72	1.05	0.88	Pink noise	MRFA
D	d_b.sh1	V	731	-	N	N	0.77	1.07	0.87	Pink noise	MRFA
D	Ec_unem	V	574	-	N	N	1.00	1.58	0.23	Brownian noise	MRFA
D	US_B_15	V	365	-	N	N	0.09	0.08	0.65	RNN	RNN
D	US_B_78	V	365	-	N	N	0.23	0.29	1.21	Anti-correlated	LSTM
H	LNOxEm	0	8081	24	Y	N	0.39	0.37	0.45	Anti-correlated	ARIMA

Sample entropy of values close to zero indicated high levels of self-similarities, and thus higher probability of predictability. For US_B_78, d_b.sh4, d.t.f.b, and d_m.t the sample entropies were relatively high and the best performance were exhibited by ARIMA, NN and LSTM. It can be seen that best performance for Brownian noise was MRFA (two series) and SVR (one series), and for pink noise series, MRFA (two series) and ARIMA (one series).

However, for the Brownian noise, SVR had the best performance on d_f.ex_r where the sample entropy was extremely small, implying that the series was highly self-similar and predictable. Also, for the Brownian noise, ARIMA came out best on d_m.t where its sample entropy was high, meaning that the complexity is high and thus the conclusion in this case is unreliable. It can be seen that ARIMA and NN outperformed the other methods when applied to stationary time series. It suggests that ARIMA and NN which are characterized by relatively less complex structures (compared to RNN, LSTM and MRFA) are predominantly suitable for series which have a steady mean, variance and auto-correlation. However, for two out of the three stationary time series, the

sample entropy was high, suggesting that stationarity can be as important as self-similarity in predictability of time series data. The results show that when Gaussianity is present MRFA was the preferred option.

For series characterized by CH, NN (m_n_o_e, GerCent, and d_t_f_b), MRFA (m_l_e_l, H_Um_3P and IreCent), SVR (S_Um_2P and d_f_ex_r) and ARIMA (d_b_sh4) preformed best. However, the series on which ARIMA was the preferred candidate, was characterized by substantial degree of White noise (Hurst exponent $\simeq 0.5$).

4.2 Analysis

It is apparent that differing methods performance are a function of the characteristics of the time series in question. MRFA does seem to posses more robust characteristics as it is either the preferred method or was one of the high performers on the majority of the series examined with a preferred candidate score of 50%. In particular it was the preferred method on 80% of the series when applied to series with either Brownian or pink noise and scored 44% when applied to data with non-stationary. MRFA, NN and LSTM all appeared to perform well when non-stationarity or conditional hetroscedasticity existed within the data. Additionally, ARIMA also performed well with non hetroscedastic data, and is much easier method to implement in comparison to machine learning approaches such as MRFA, NN, RNN or LSTM. Fig. 1 outlines the performance of all the methods attempted.

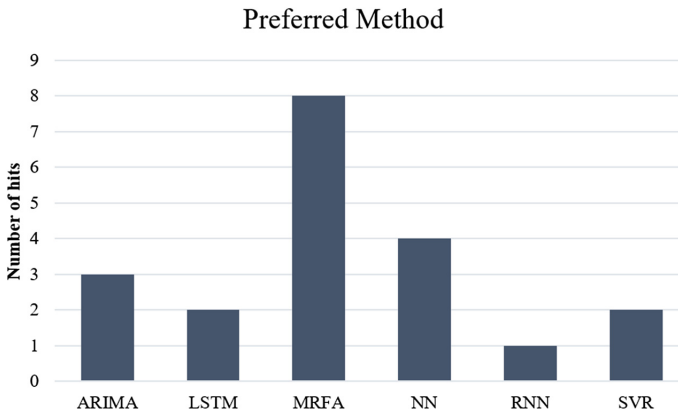


Fig. 1. Method versus preferred method count

5 Conclusions

This paper proposes an analysis that will assist researchers in the selection of the most appropriate prediction model based on historical meta analysis of

time series characteristics. A number of important characteristics were analyzed including frequency, sample size, Gaussianity, long-term memory (Hurst exponent and Detrended Fluctuation Analysis), non-stationarity, and conditional heteroscedasticity. Experimental results were obtained on 20 series with varying characteristics including anti-correlated series, pink noise, Brownian noise, stationary and non-stationary series. The evaluation demonstrates that the performance of a prediction method, in particular machine learning techniques, is a function of the characteristics of the given time series. Using a meta analysis of historical time series analysis as a basis for the selection of a time series approach will reduce the effort required within each specific application. Additionally, increasing the size of the Meta analysis database substantially would have obvious benefits to researchers in areas such as the Agri or Finance sectors.

References

1. Aweya, J.: Sensitivity methods for congestion control in computer networks. Ph.D thesis, Ottawa, Ontario, Canada, AAINQ48085 (1999)
2. Bahrpeyma, F., Roantree, M., McCarren, A.: Multi-resolution forecast aggregation for time series in agri datasets. In: Proceedings of the 25th Irish Conference on Artificial Intelligence and Cognitive Science, Dublin, Ireland, 7–8 December 2017, pp. 193–205 (2017)
3. Bontempi, G., Ben Taieb, S., Le Borgne, Y.-A.: Machine learning strategies for time series forecasting. In: Aaufaure, M.-A., Zimányi, E. (eds.) eBISS 2012. LNBP, vol. 138, pp. 62–77. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36318-4_3
4. Box, G.E.P., Jenkins, G.M., Reinsel, G.C., Ljung, G.M.: Time Series Analysis: Forecasting and Control. Wiley, Hoboken (2015)
5. Brockwell, P.J., Davis, R.A.: Introduction to Time Series and Forecasting. Springer, Heidelberg (2016). <https://doi.org/10.1007/978-3-319-29854-2>
6. Browne, A.: Neural Network Analysis, Architectures and Applications. CRC Press, Boca Raton (1997)
7. Cadavid, A.C., Lawrence, J.K., Ruzmaikin, A.: Principal components and independent component analysis of solar and space data. In: Ireland, J., Young, C.A. (eds.) Solar Image Analysis and Visualization. Springer, New York (2007). https://doi.org/10.1007/978-0-387-98154-3_5
8. Chatfield, C.: The Analysis of Time Series: An Introduction. CRC Press, New York (2016)
9. Chu, H., Wei, J., Li, T., Jia, K.: Application of support vector regression for mid- and long-term runoff forecasting in “yellow river headwater” region. *Procedia Eng.* **154**, 1251–1257 (2016)
10. Corder, G.W., Foreman, D.I.: Nonparametric Statistics: A Step-by-Step Approach. Wiley, Hoboken (2014)
11. Dickey, D.A., Fuller, W.A.: Distribution of the estimators for autoregressive time series with a unit root. *J. Am. Stat. Assoc.* **74**(366a), 427–431 (1979)
12. Fischer, T., Krauss, C.: Deep learning with long short-term memory networks for financial market predictions. *Eur. J. Oper. Res.* **270**, 654–669 (2017)
13. Hurst, H.E.: Long term storage capacity of reservoirs. *ASCE Trans.* **116**(776), 770–808 (1951)

14. Kantelhardt, J.W., Koscielny-Bunde, E., Rego, H.H.A., Havlin, S., Bunde, A.: Detecting long-range correlations with detrended fluctuation analysis. *Phys. A Stat. Mech. Appl.* **295**(3–4), 441–454 (2001)
15. Kočenda, E., Černý, A.: *Elements of Time Series Econometrics: An Applied Approach*. Charles University in Prague, Karolinum Press, Prague (2015)
16. Kwiatkowski, D., Phillips, P.C.B., Schmidt, P., Shin, Y.: Testing the null hypothesis of stationarity against the alternative of a unit root: how sure are we that economic time series have a unit root? *J. Econom.* **54**(1–3), 159–178 (1992)
17. Mandic, D.P., Chambers, J.A.: *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*. Wiley Online Library (2001)
18. Parlos, A.G., Rais, O.T., Atiya, A.F.: Multi-step-ahead prediction using dynamic recurrent neural networks. *Neural Netw.* **13**(7), 765–786 (2000)
19. Richman, J.S., Lake, D.E., Moorman, J.R.: Sample entropy. In: *Methods in Enzymology*, vol. 384, pp. 172–184. Elsevier (2004)
20. Soofi, A.S., Cao, L.: *Modelling and Forecasting Financial Data: Techniques of Nonlinear Dynamics*, vol. 2. Springer, New York (2012). <https://doi.org/10.1007/978-1-4615-0931-8>
21. Xiong, W., Xu, B.: Study on optimization of SVR parameters selection based on PSO. *J. Syst. Simul.* **9**, 017 (2006)
22. Zhang, G., Hu, M.Y.: Neural network forecasting of the British pound/US dollar exchange rate. *Omega* **26**(4), 495–506 (1998)



Novel Data Segmentation Techniques for Efficient Discovery of Correlated Patterns Using Parallel Algorithms

Amulya Kotni¹(✉), R. Uday Kiran^{2,3}, Masashi Toyoda², P. Krishna Reddy¹,
and Masaru Kitsuregawa^{2,4}

¹ International Institute of Information Technology, Hyderabad, India
kotni.amulya@research.iiit.ac.in, pkreddy@iiit.ac.in

² Institute of Industrial Science, The University of Tokyo, Tokyo, Japan
{uday_rage,toyoda,kitsure}@tkl.iis.u-tokyo.ac.jp

³ National Institute of Information and Communication Technologies, Tokyo, Japan

⁴ National Institute of Informatics, Tokyo, Japan

Abstract. Efficient discovery of interesting patterns using parallel algorithms is an actively studied topic in data mining. A key research issue related to this topic is data segmentation, which influences the overall computational requirements of an algorithm. This paper makes an effort to address this issue in correlated pattern mining. Two novel data segmentation techniques, ‘database segmentation’ and ‘transaction segmentation,’ have been introduced to discover the patterns efficiently. The former technique involves segmenting the database into multiple sub-databases such that each sub-database can be mined independently. The latter technique involves segmenting a transaction into multiple sub-transactions such that each sub-transaction can be processed as an individual transaction. The proposed techniques are algorithm independent, and therefore, can be incorporated into any parallel algorithm to find correlated patterns effectively. In this paper, we introduce map-reduce based pattern-growth algorithm by incorporating the above mentioned techniques. Experimental results demonstrate that the proposed algorithm is memory and runtime efficient and highly scalable as well.

Keywords: Data mining · Knowledge discovery in databases
Parallel algorithms · Correlated patterns and map-reduce

1 Introduction

Frequent pattern mining is an important model in data mining. Its mining algorithms discover all patterns in the data that satisfy the user-specified *minimum support* (*minSup*) constraint [1, 2]. The popular adoption and successful industrial application of this model has been hindered by the following obstacle: *the frequent pattern model involves exponential mining space and often generates a huge number of patterns*. To confront this problem, researchers have introduced

correlated pattern mining [3]. It is because the users may be interested in not only the frequent occurrences of sets of items, but also their possible strong correlations implied by such co-occurrences. The usefulness of correlated patterns was demonstrated in many applications such as climate studies, public health and bioinformatics.

In the literature, researchers have discussed several measures to assess the interestingness of a pattern. Examples include *all-confidence* [4], *any-confidence* [4], *lift* [3], *bond* [4], *h-confidence* [5] and *relative support* [6]. Each measure has a selection bias that justifies the significance of a knowledge pattern. As a result, there exists no universally acceptable best measure to judge the interestingness of a pattern for any given database or application. Researchers are making efforts to suggest a right measure depending upon the user and/or application requirements [7, 8].

Recently, *all-confidence* is emerging as a popular measure to discover correlated patterns [9–11]. It is because this measure satisfies both the *anti-monotonic* and *null-invariance* properties. The former property says that “all non-empty subsets of a correlated pattern must also be correlated.” This property plays a key role in reducing the search space, which in turn decreases the computational cost of mining the patterns. In other words, this property makes the pattern mining practicable in real-world applications. The latter property discloses genuine correlation relationships without being influenced by the object co-absence in a database. In other words, this property facilitates the user to discover interesting patterns involving both frequent and rare items without generating a huge number of uninteresting patterns. In this paper, we focus on finding correlated patterns using the *all-confidence* measure.

The basic model of correlated patterns is as follows [9]: Let $I = \{i_1, i_2, \dots, i_n\}$, $n \geq 1$, be a set of items, and TDB be a database that consists of a set of transactions. Each transaction T contains a set of items such that $T \subseteq I$. Each transaction is associated with an identifier, called TID . Let $X \subseteq I$ be a set of items, referred as an itemset (or a pattern). A pattern that contains k items is a k -pattern. A transaction T is said to contain X if and only if $X \subseteq T$. The *support* of a pattern X in TDB , denoted as $S(X)$, is the number of transactions in TDB containing X . The pattern X is a **frequent pattern** if $S(X) \geq minSup$, where *minSup* represents the user-specified *minimum support*. The *all-confidence* of a pattern X , denoted as *all-conf*(X), can be expressed as the ratio of its *support* to the *maximum support* of an item within it. That is, $all-conf(X) = \frac{S(X)}{\max(S(i_j) | \forall i_j \in X)}$. A pattern X is said to be *all-confident* or *associated* or *correlated* if $S(X) \geq minSup$ and $all-conf(X) \geq minAllConf$, where *minAllConf* represents the user-specified minimum all-confidence.

Example 1. Consider the transactional database shown in Table 1. The set of items $I = \{a, b, c, d, e, f, g, h, i, j\}$. The set of items ‘ a ’ and ‘ b ,’ i.e., $\{a, b\}$ (or ab) is a pattern. This pattern contains 2 items. Therefore, it is a 2-pattern. The pattern ab occurs in 7 transactions. Henceforth, the support of ‘ ab ,’ i.e., $S(ab) = 7$. If the user-specified $minSup = 3$, then ‘ ab ’ is a frequent pattern because $S(ab) \geq minSup$. The *all-confidence* of ‘ ab ,’ i.e., $all-conf(ab) = \frac{S(ab)}{\max(S(a), S(b))} =$

$\frac{7}{\max(10,9)} = 0.7$. If the user-specified $minAllConf = 0.7$, then the frequent pattern ‘ ab ’ is a correlated pattern because $all-conf(ab) \geq minAllConf$.

Lee et al. [9] discussed a pattern-growth algorithm, called CoMine, to find all correlated patterns in a transactional database. Uday et al. [12] have described an improved CoMine algorithm based on the property of *items’ support intervals*. This property says that each item in the database can generate correlated patterns of higher order by combining with only those items that have support within a particular interval. Both CoMine and CoMine++ are sequential algorithms, and do not exploit the availability of multiple machines or the presence of multiple cores in a machine for computation.

Table 1. Transactional database

TID	Items	TID	Items	TID	Items
1	abcde	5	abcde	9	abfg
2	ace	6	ab	10	aj
3	abfgh	7	bcdi	11	abcd
4	abf	8	adj	12	bfg

A Map-Reduced framework to exploit the power of thousands of machines is proposed in [13]. Encouraged by the power of Map-Reduce paradigm, researchers are making efforts to propose parallel algorithms to find frequent patterns under Map-Reduce framework [14, 15]. These parallel frequent pattern mining algorithms can be extended to mine correlated patterns. However, it was revealed in our investigation that such naive algorithms are inefficient because they do not take into account the properties of *all-confidence* measure. In this paper, we make an effort to develop an efficient parallel correlated pattern mining algorithm by utilizing the properties of *all-confidence* and Map-Reduce framework.

A key research issue in developing an efficient parallel algorithm is the data segment, which influences the overall computational cost of mining the correlated patterns. Two novel data segmentation techniques, ‘database segmentation’ and ‘transaction segmentation,’ have been introduced to address this issue. The former technique involves grouping the items with respect to their *supports* and splitting the whole database into multiple sub-databases such that each sub-database can be mined independently without missing any correlated pattern. This technique enables us to efficiently distribute data across multiple machines. The second technique involves splitting a transaction into multiple sub-transactions such that each sub-transaction can be treated and processed as an independent transaction. Both techniques are based on the concept of *items’ support intervals*, and are independent of the mining algorithm. Using the proposed techniques and Map-Reduce framework, we introduce a parallel algorithm, called Parallel Correlated Pattern-growth (PCP-growth), to find correlated patterns. Experimental results demonstrate that PCP-growth is memory and runtime efficient.

The rest of the paper is organized as follows. Section 2 describes the related work on parallel algorithms and correlated pattern mining. Section 3 introduces the proposed data segmentation techniques. Section 4 describes the proposed PCP-growth algorithm. Experimental results are reported in Sect. 5. Finally, Sect. 6 concludes the paper with future research directions

2 Related Work

2.1 Frequent Pattern Mining

Since the introduction of frequent patterns, several algorithms have been discussed to find these patterns efficiently [1, 2, 14, 15]. Two popular frequent pattern mining algorithms are Apriori [1] and Frequent Pattern-growth (FP-growth) [2]. Apriori employs a candidate-generate-and-test approach to find frequent patterns, while FP-growth employs pattern-growth approach to find frequent patterns. It has been argued in the literature that FP-growth is typically better than the Apriori because the latter suffers from the performance issues such as generating huge number of candidate patterns and requiring multiple scans on the database. Both Apriori and FP-growth are sequential algorithms that suffer from memory issues when employed on very large databases.

Pramudiono et al. [16] proposed a distributed FP-Growth algorithm by taking into account a cluster of machines. Li et al. [14] proposed a Parallel FP-Growth algorithm (PFP-growth) using Map-Reduce framework. Xun et al. [15] improved the performance of PFP-growth by compressing the data into FUI-tree rather than the FP-tree. This algorithm is known as FiDooP. It has to be noted that all of the above mentioned algorithms focuses on finding frequent patterns using *minSup*. These algorithms can be extended to find correlated patterns using all-confidence measure. However, such naive algorithms are inefficient as they do not take into account the properties of all-confidence measure.

2.2 Correlated Pattern Mining

Brin et al. [3] introduced correlated pattern mining using *lift* and χ^2 as the interestingness measures. Lee et al. [9] have shown that correlated patterns can be effectively discovered with all-confidence measure as it satisfies both null-invariance and downward closure properties. An FP-growth-like algorithm, called CoMine, was discussed to find the patterns. A variant of CoMine, CCMine [10], was proposed to discover confidence-closed correlated patterns. Kim et al. [11] have made an effort to discover top-k correlated patterns using the measures that satisfy the null-invariance property. Since some of the null-invariant measures (e.g. cosine) do not satisfy the anti-monotonic property, an Apriori-like algorithm was discussed to find the patterns. Uday et al. [12] discussed CoMine++ based on the concept of items' *support* intervals. This concept will be discussed in latter parts of this paper. All of the above mentioned algorithms are sequential algorithms. This paper focuses on developing parallel correlated pattern mining algorithm.

In the next section, we describe novel data segmentation techniques for distributing data across multiple machines in a network efficiently.

3 Proposed Data Segmentation Techniques

3.1 Items' Support Intervals

Uday et al. [12] introduced the property of items' support intervals to find correlated patterns effectively. This property is based on Apriori property (See Property 1) [1] and is defined as follows:

Property 1. Apriori Property. If $X \subset Y$, then $S(X) \geq S(Y)$.

Definition 1. *The support interval of an item $i_j(SI(i_j))$. An item $i_j \in I$ can generate correlated patterns of higher order by combining with only those items that have support within the interval $\left[\max(S(i_j) \times \min AllConf, \min Sup), \max\left(\frac{S(i_j)}{\min AllConf}, \min Sup\right) \right]$. Thus, the support interval of i_j , denoted as $SI(i_j) = \left[\max(S(i_j) \times \min AllConf, \min Sup), \max\left(\frac{S(i_j)}{\min AllConf}, \min Sup\right) \right]$.*

The correctness of this property is given in [12]. Example 2 illustrates this property.

Example 2. In Table 1, the item 'c' has support of 5. If the user-specified $\min Sup = 3$ and $\min AllConf = 0.7$, then the support interval of 'c' is $[4, 7]$ ($= [\max(3.5, 3), \max(7, 3)]$). It means 'c' can generate correlated patterns of higher order by combining with only those items that have support within the interval $[4, 7]$. If 'c' combines with items having support not within its support interval, then Apriori property [1] causes the resulting pattern to have $allConf < \min AllConf$ or $sup < \min Sup$ or both. For instance, 'c' cannot generate correlated pattern by combining with 'b', whose support of 9 doesn't lie within the support interval of 'c'. The reason is, $S(bc) \leq \min Sup$ (Apriori Property), and thus $allConf(bc) < \min AllConf$. The support intervals of all frequent items in Table 1 are shown in Table 2.

We now introduce data segmentation techniques based on the above mentioned items' support intervals. Please note that the proposed techniques are novel to this paper and have not been used in CoMine++ algorithm [12].

Table 2. The support intervals (SI) and group ids (GI) of all frequent items in Table 1

Item	Support	SI	GI	Item	Support	SI	GI
a	10	[7,14]	1	f	4	[3,5]	2
b	9	[7,12]	1	e	3	[3,4]	2
c	5	[4,7]	2	g	3	[3,4]	2
d	4	[3,5]	2				

3.2 Data Segmentation Techniques

Data segmentation plays a key role in parallel algorithms. It influences: (i) overall memory and runtime requirements of an algorithm and (ii) communication cost across the machines. We now describe two data segmentation techniques to discover correlated patterns effectively.

1. Database Segmentation. In the database segmentation, we initially group the items such that items within a group will not generate correlated patterns by combining with the items in other groups. Next, we split the given database into sub-databases such that each sub-database contains items of a specific group. This approach of splitting the database facilitates us to mine each sub-database independently to find correlated patterns. If there are m groups of items, then there will be m sub-databases as each sub-database contains items of a particular group. The m value depends on the distribution of items' *support* values and the user-specified *minAllConf* and *minSup* values. We now describe grouping of items.

Algorithm 1. DatabaseSegmentation($TDB, minSup, minAllConf$)

- 1: Scan the database to determine the *support* and support interval of items. Next, sort all items in descending order of their *support*. Let I denote the sorted list of items. Let $S(i_p)$ denote the *support* of an item i and p representing its position (or rank) in the list. $SI_{i_p}[a, b]$, $a, b \in \mathbb{R}$, denote the support interval of item i_p . Let $G(i_p)$ denote the group id of item i_p .
 - 2: Set $group_id = 1$ and $G(i_0) = 1$.
 - 3: **for** $p = 0; p < I.size() - 1; ++p$ **do**
 - 4: **if** $S(i_p) < SI_{i_{p+1}}.b$ **then**
 - 5: set $G(i_{p+1}) = group_id$.
 - 6: **else**
 - 7: $++group_id$;
 - 8: $G(i_{p+1}) = group_id$.
 - 9: Split the database with respect to items' group.
-

Definition 2. Group of items (GI). Let $I = \{i_1, i_2, \dots, i_n\}$, $n \geq 1$, be the set of sorted items such that $S(i_1) \geq S(i_2) \geq \dots \geq S(i_n)$. Let $GI \subseteq I$ be a maximal set of items such that the support of an item $i_p \in GI$ lies within the support interval of its subsequent item in GI . That is, $GI \subseteq I$, such that if $i_p, i_{p+1} \in GI$, then $S(i_p) \in SI(i_{p+1})$, $1 \leq p < (|GI| - 1)$.

Example 3. In Table 2, it can be observed that the *support* of 'a' lies within the support interval of 'b', whereas b's *support* does not lie within the support interval of c. So we consider 'a' and 'b' as one group of items. Similarly, we consider 'c', 'd', 'e', 'f' and 'g' as another group of items. Thus, the set of frequent items in Table 1 can be divided into two disjoint groups, i.e., $GI_1 = \{a, b\}$ and

$GI_2 = \{c, d, e, f, g\}$. (For brevity, we have not considered infrequent items in Table 1).

Definition 3. Sub-database (SDB). Let GI_1, GI_2, \dots, GI_m be the set of disjoint group of items such that $GI_1 \cup GI_2 \cup \dots \cup GI_m = I$ and $GI_p \cap GI_q = \emptyset$, $p \neq q$ and $p, q \in [1, m]$. Let $SDB_i \subseteq TDB$, $1 \leq i \leq m$, be the sub-database such that all items in SDB_i belong to a particular group GI_i .

Example 4. Continuing with the previous example, the items’ groups provide useful information that ‘a’ and ‘b’ will generate correlated patterns by combining between themselves only. The same can be said about the remaining items. Thus, the given database can be divided into two sub-databases as shown in Tables 3 and 4. Each of these sub-databases can be mined independently to discover correlated patterns.

The two main advantages of database segmentation are: (i) we can eliminate mining of correlated patterns in sub-databases containing a single item and (ii) If a sub-database contains two items, a single scan on the entire database finds all correlated patterns. The algorithm for database segmentation is given in Algorithm 1. As the algorithm is straight forward to understand, we are not describing the algorithm.

Table 3. Sub-database 1

<i>id</i>	transaction	<i>id</i>	transaction	<i>id</i>	transaction
1	ab	5	ab	9	ab
2	a	6	ab	10	a
3	ab	7	b	11	ab
4	ab	8	a	12	b

Table 4. Sub-database 2

<i>id</i>	transaction	<i>id</i>	transaction	<i>id</i>	transaction
1	ce	5	cde	9	fg
2	ce	6	–	10	–
3	fg	7	cd	11	cd
4	f	8	d	12	fg

2. Transaction Segmentation. After performing database segmentation, we have distributed sub-databases to various machines to discover correlated patterns. (We used a variant of PFP-growth algorithm [14] to find correlated patterns in various machines.) During this process, we have observed that some machines were taking more time to output correlated patterns. Our investigation on the cause has revealed that the sub-databases being executed in these machines actually contained long transactions, and processing these long transactions increased the runtime. Therefore, we introduced transaction segmentation to reduce the runtime requirements of a parallel algorithm.

Definition 4. Transaction segmentation. Let $T = \{i_1, i_2, \dots, i_m\} \subseteq I$, $1 \leq m \leq n$, be a transaction in a (sub-)database such that $S(i_1) \geq S(i_2) \geq \dots \geq S(i_m)$. A sub-transaction $T_1 = \{i_a, i_{a+1}, \dots, i_b\} \subseteq T$, $1 \leq a \leq b \leq m$ is a **maximal set of items** such that $S(i_a) \in SI(i_{a+1})$. Thus, $T = T_1 \cup T_2 \cup \dots \cup T_p$, $p \geq 1$ and $T_x \cap T_y = \emptyset$, $x, y \in [1, p]$ and $x \neq y$.

Example 5. Consider the first transaction ‘ce’ in Table 4. In this transaction, the support of ‘c’ does not lie within the support interval of its neighboring item ‘e’. Henceforth, this transaction can be further segmented though the items in this transaction belong to the same group. In other words, the transaction ‘ce’ can be split into two independent sub-transactions, ‘c’ and ‘e’. Please note that the fifth transaction ‘cde’ in Table 4 will not be segmented because c’s support lies within the support interval of its neighboring item ‘d’, and d’s support lies within the support interval of its neighboring item ‘e’.

The procedure for transaction segmentation is given in Algorithm 2. Since the algorithm is straight forward to understand, we are not describing the algorithm.

Both techniques are algorithm independent, and therefore, can be used in any parallel algorithm (such as Apriori and FP-growth). More importantly, the proposed techniques are network independent and therefore can be used in cluster computing, grid computing and Map-Reduce framework. In this paper, we extend the proposed techniques to develop efficient algorithm for Map-Reduce framework.

4 Proposed Algorithm: PCP-growth

The PCP-growth algorithm employs Map-Reduce framework to discover correlated patterns effectively. The algorithm involves the following two steps: (i) finding frequent items and their support intervals and (ii) mining all correlated patterns by constructing FP-trees. We now discuss each of these steps.

Algorithm 2. TransactionSegmentation(*SDB*: sub-database, *SI*: support intervals of items)

```

1: Output: ST=[] be the list of all sub-transactions
2: Let  $p = \text{NULL}$ 
3: for each transaction  $t$  in  $SDB$  do
4:   Sort the items in the transaction in descending order of supports.
5:   for  $q=0; q < t.size()-1; ++q$  do
6:      $p += i_q$ 
7:     if  $S(i_q) > SI_{i_{q+1}}.b$  then
8:       ST.append( $p$ ) //Transaction  $t$  is split here and the sub-transaction  $p$  is
         added to ST
9:      $p = \text{NULL}$ 
10:     $p += i_{q+1}$ 
11:    ST.append( $p$ )

```

Finding Frequent Items and Their Support Intervals: First, the transactional database is divided into multiple shards (partitions) and provided to worker machines. The number of shards generally depend on the available number of machines. In the map phase, each worker machine scans the transactions one after another. For each transaction, worker machine outputs key-value pairs with key as the item and value as 1 (format is $\langle item, 1 \rangle$). In the reduce phase, all the pairs with the same key obtained from all the worker machines are aggregated. Thus, supports for all the items present in the database are calculated. The final list of frequent items, called FP-list, is obtained by filtering out the items which do not satisfy the $minSup$ threshold. After the master machine obtains the FP-list, the items are sorted in decreasing order of their support values and assigned ranks. The most frequent item is assigned a rank of 0, the second most frequent item is assigned a rank of 1 and so on. Next, the support intervals for the frequent items is calculated according to Definition 1. Next, items are assigned group ids such that items within a group have support within the support interval of the subsequent item. These group ids are later used for database segmentation. After this step, the master node broadcasts the items' support interval information to all the worker nodes in order to facilitate transaction segmentation in the next step (Fig. 1).

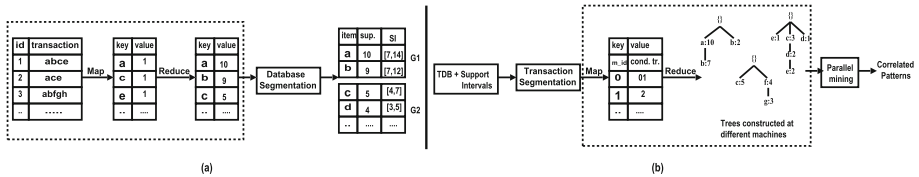


Fig. 1. Stages of PCP-Growth algorithm. (a) Assigning group ids to the items and (b) mining the patterns

Algorithm 3. Parallel FP-list Construction($TDB, minSup$)

- 1: **Procedure:** Map($key = null, value = TDB_i$)
 - 2: **for** each transaction $t_{cur} \in TDB_i$ **do**
 - 3: **for** each item it in t_{cur} **do**
 - 4: Output ($it, 1$)
 - 1: **Procedure:** Reduce($key = it, value = 1$)
 - 2: $sup = 0$
 - 3: **for** each it **do**
 - 4: $sup++$
 - 5: Output (it, sup)
 - 6: **if** $sup \geq minSup$ **then**
 - 7: Add (it, sup) to FP-list
-

Example 6. The transactional database shown in Table 1 is divided into 3 partitions and provided to worker machines. These worker machines determine the *support* values for all items using Map-Reduce framework. Next, frequent items are generated by removing all items whose support is less than the user-specified *minSup*. The remaining frequent items are sorted in descending order of their supports. Next, the support intervals are calculated using the Definition 1. The items are now divided into groups using items' support intervals (Database Segmentation). The support intervals and group ids for all the frequent items are shown in Table 2.

Construction and Mining of FP-trees: The number of machines available in the distributed network are divided according to the ratio of number of items present in the groups. In the second database scan, for each transaction the items which are not present in the FP-list are filtered and the remaining items are sorted in descending order of supports. Now, using the transaction segmentation technique discussed in Sect. 3.2, we split the transaction into smaller transactions. The mining task of each item present in an item group is assigned to one of the machines assigned to that group. The sub-patterns (conditional transactions) for each sub-transaction are extracted and assigned to a machine based on a hash function. Here, item is the last item in a sub-pattern. Here, the assigned machine is stored in a hash-table for future look-up. The hash function gives a machine-id for which the pattern is responsible for further computation. The sub-patterns for every sub transaction are generated and sent to the corresponding machine. Each sub-pattern is emitted as a key-value pair, with key as the machine-id and value as sub-pattern. If a machine is responsible for many sub-patterns of the same transaction, only the longest sub-pattern is sent because the others can be derived from it. Now, the reduce function is implemented with machine-id as key, hence all the conditional transactions with same partition-id are processed at one machine. Independent local FP-trees are constructed by inserting all the sub-patterns into the tree in the same order as FP-list. The process of tree construction is same as FP-tree construction in [2]. Trees constructed on three different machines are shown in Fig. 2. Since the trees are constructed from the sub-patterns itself, during conditional pattern building, communication is not required between the machines.

Example 7. The frequent items generated in the previous step (see Table 2) are assigned ranks in descending order of their *support*. That is, the most frequent item 'a' is assigned with rank 0, 'b' is assigned with rank 1 and so on. Consider the first transaction 'abce' in the first partition. Upon removing the infrequent items and sorting the remaining items in decreasing order of their supports, the transaction is converted into 'abce'. First, we perform database segmentation and split this transaction into two sub-transactions 'ab' and 'ce' such that each sub-transaction belongs to a sub-database. In the sub-transaction 'ab' the support of 'a' lies in the support interval of 'b'. Therefore, without performing transaction segmentation on this transaction, conditional patterns are generated and sent to the machine(s) responsible for GI_1 . Now, let us consider the sub-transaction

Algorithm 4. PCP-Growth Algorithm(TDB, FP-list, SI)

```

1: Allocate machines in the distributed network to the Item Groups.
2: Procedure: Map(key = null, value = TDBi)
3:   for each transaction  $t_{cur} \in TDB_i$  do
4:     Filter the items that are not in FP-list and sort them.
5:     Segment the transaction based on SI values. Let  $T'$  be the set of sub-
        transactions obtained from  $t_{cur}$ .
6:     for each sub-transaction  $t'$  in  $T'$  do
7:       for each item  $i'$  in  $t'$  do
8:         machine-id = getMachineId( $i'$ )
1: Procedure: Reduce(key = machine-id, value = transactions)
2:   Initialize FP-Tree, T
3:   for  $t_{cur}$  in transactions do
4:     for item in  $t_{cur}$  do
5:       if T does not have child item then
6:         Create a new node and link it to the parent node
7:         Traverse to the child item
1: Procedure: Map(key = machine-id, value = FP-Tree)
2:   for each suffix item  $i$  in FP-Tree do
3:     if current machine-id is responsible for item  $i$  then
4:       Generate prefix tree and conditional tree of  $i$  and mine recursively.

```

‘ce’. In this sub-transaction although the items ‘c’ and ‘e’ belong to the same group, the support of ‘c’ does not lie in the support interval of ‘e’(or vice versa). Therefore, we perform transaction segmentation and further split ‘ce’ into sub-transactions ‘c’ and ‘e’.

Let the 3 machines in the distributed network be m_0 , m_1 and m_2 . These machines are distributed among the item groups GI_1 and GI_2 . Assume m_0 is allocated to GI_1 and m_1 and m_2 are allocated to GI_2 . The sub-transactions for the transaction ‘abce’ are ‘ab’, ‘c’ and ‘e’. Upon translating the items in the sub-transactions into their ranks, they are converted into ‘01’, ‘2’ and ‘5’. Now, for each sub-transaction, sub-patterns (conditional patterns) are generated and assigned to the machine responsible for it using a hash function. Two sub-patterns are generated for the sub-transaction ‘01’, which are ‘0’ and ‘01’. As GI_1 has only one machine allocated to it, all the sub-patterns ending with ‘0’ or ‘1’ are hashed to m_0 . So, the key-value pairs outputted are {0:‘0’, 0:‘01’}. Both the sub-patterns are assigned to the same machine, so only the longest sub-pattern ‘01’ is sent as ‘0’ can be derives from it. For the sub-transaction ‘2’, one sub-pattern ‘2’ is generated. As item ‘c’ belongs to item group GI_2 and two machines are allocated to it, the sub-pattern ‘2’ is hashed to either m_1 or m_2 using a hash function. Similarly, for the sub-transaction ‘5’, one sub-pattern ‘5’ is generated and is hashed to either m_1 or m_2 . Following conditional transactions are received at the m_0 , { $ab, a, ab, ab, ab, ab, b, a, ab, a, ab, b$ }. As per the output of the hash function used, the items ‘c’, ‘f’ and ‘g’ are assigned to m_1 and the items

'd' and 'e' are assigned to m_2 . The following conditional transactions are received at m_1 , $\{c, c, fg, f, c, c, fg, c, fg\}$. Similarly, m_2 receives the following conditional patterns, $\{e, e, cde, cd, d, cd\}$.

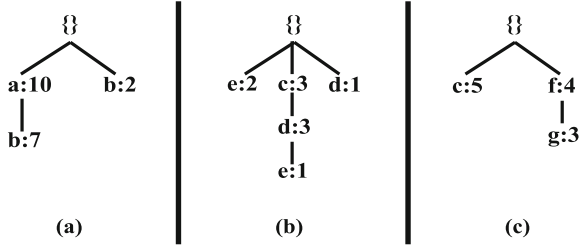


Fig. 2. (a) Tree at m_0 . (b) Tree at m_1 . (c) Tree at m_2

Parallel mining of correlated patterns is similar to the mining process of FP-Growth algorithm but each worker machine performs the mining process only for those suffix items for which it is responsible for computation. The prefix tree is constructed for a chosen suffix item by inserting the prefix sub paths of the nodes of the selected item. The conditional tree is constructed from the prefix tree by removing the nodes which do not satisfy the *minSup* threshold. This process is repeated for all the items assigned to each worker node. Now, all the items are translated back into their original values. Finally, the patterns extracted by all the worker nodes are gathered at the master node. Consider the mining of patterns with suffix items 'g' and 'c,' as shown in Fig. 3. In the existing sequential approach, the mining process of 'c' occurs only after the mining process of 'g' and other items below 'c' in the FP-list is completed. Whereas in the proposed approach, both the processes happen in parallel.

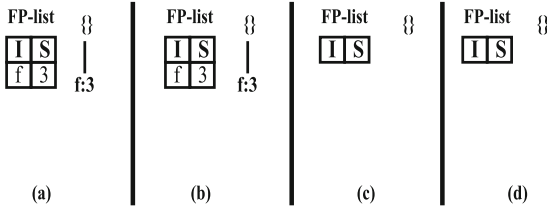


Fig. 3. Mining for suffix items 'g' and 'c' (a) Prefix tree of 'g' (b) Conditional tree of 'g' (c) Prefix tree of 'c' (d) Conditional tree of 'c'

Example 8. The mining process is started from the least frequent item, which is 'g' in the above example. The item 'g' is assigned to m_2 . So, the mining of

patterns with suffix ‘ g ’ is done at the tree built at m_2 i.e., Fig.2(c). The prefix tree for ‘ g ’ is constructed from this tree by removing the nodes of item ‘ g ’. There is only one branch $\langle f : 3 \rangle$ containing the item g . The final prefix tree for item ‘ g ’ is shown in Fig.3(a). From the prefix tree, conditional tree is constructed by removing the items which do not satisfy the $minSup$ threshold. The item ‘ f ’ satisfies the $minSup$ threshold. The conditional tree for item ‘ g ’ is shown in Fig.3(b). Thus, from this tree the correlated patterns obtained are $\{g : 3, fg : 3\}$. Consider the item ‘ c ’. For mining th patterns with suffix ‘ c ’, the tree at m_1 has to be mined as ‘ c ’ is assigned to it. Though the tree at m_2 has nodes containing item ‘ c ’, it need not be mined as the mining of ‘ c ’ will be done in the tree at m_1 . The prefix tree of ‘ c ’ is constructed from the tree in Fig.2(b). The prefix tree of ‘ c ’ is NULL, so the conditional tree is also NULL, as shown in Fig.3(c) and (d) respectively. Therefore, the only correlated pattern generated is $\{c : 5\}$. This mining process is repeated for all the items in their respective trees and the final correlated patterns are obtained.

5 Experimental Results

In this section, we evaluate the performance of PCP-growth algorithm. The algorithm is written in Python using Apache Spark architecture and the experiments are conducted on Amazon Elastic Map-Reduce cluster, with each machine having 8 GB memory. The runtime is measured in seconds and specifies the total execution time of the spark job. We conducted the experiments on both synthetic (T10I4D100K) and real-world (Retail and Online Store [17]) datasets. The T10I4D100K dataset contains 100,000 transactions with 870 items. The Retail dataset contains 88,162 transactions with 16,470 items. The Online store dataset contains 541,909 transactions with 2,603 items. For all the experiments, the $minSup$ threshold for these datasets is set to 0.1%. Since there is no existing parallel algorithm for finding correlated patterns, we extend the existing PFP-growth algorithm to find correlated patterns using all-confidence measure. We call this algorithm as **naive algorithm** and compare it against the proposed PCP-growth algorithm.

Figures 4(a), (b) and (c) show the number of correlated patterns generated in T10I4D100k, Retail and Online Store databsets respectively at different $minAllConf$ values. It can be observed that with increase in $minAllConf$ value, the number of correlated patterns generated decreases.

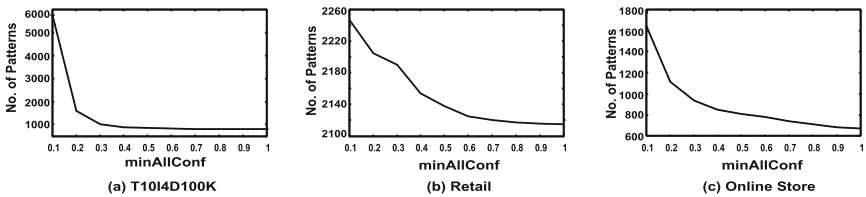


Fig. 4. Correlated patterns generated in various databases.

Figures 5(a), (b) and (c) show the number of sub-databases generated for various datasets at different $minAllConf$ values. It can be observed that increase in $minAllConf$ increases the number of sub-databases. The reason is as follows: *increase in $minAllConf$ decreases the support-interval range for the items. The reduction of support-interval ranges for the items increases the number of sub-databases.*

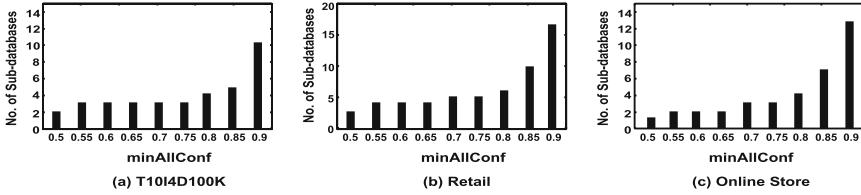


Fig. 5. Number of sub-databases formed in various databases

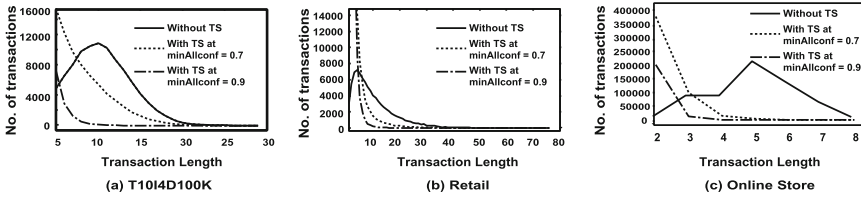


Fig. 6. Transaction length vs. Frequency of transactions for various databases

Table 5. Number of 1-length transactions. The term TS is used as an acronym for transaction segmentation

Dataset	Without TS	With TS at $minAllConf = 0.7$	With TS at $minAllConf = 0.9$
T10I4D100K	128	127,798	470,429
Retail	3016	269,114	566,261
Online Store	2635	1,238,305	1,922,804

Figures 6(a) (b) and (c) show the distribution of transaction lengths in various databases before and after applying transaction segmentation technique. Since number of singleton transactions (or transactions containing only one item) generated after transaction segmentation are too many, we have presented these results separately in Table 5. It can be observed that transaction segmentation has segmented many larger transactions into smaller transactions depending upon the $minAllConf$ value. These shorter transactions reduce the *tree* size in PCP-growth algorithm.

Figures 7(a), (b) and (c) show the variation of total time consumed with the number of machines. It can be observed that increase in number of machines decreases the runtime for both naive and proposed algorithms. However, proposed algorithm was at least 50% faster than the naive algorithm.

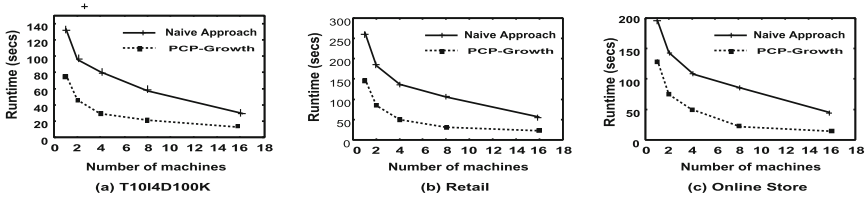


Fig. 7. Runtime vs. Number of machines for various databases

Figures 8(a), (b) and (c) show the amount of data shuffled among the machines. It can be observed that increase in machines increases the amount of data shuffled for both naive and proposed algorithms. However, it can be observed that the data shuffled is very less in PCP-growth algorithm.

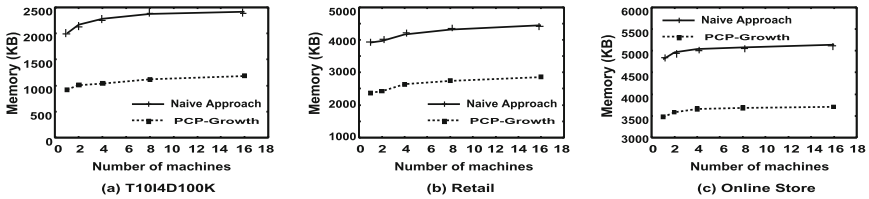


Fig. 8. Total amount of data shuffled for various databases

6 Conclusions and Future Work

Correlated pattern mining is an important model in data mining. Most of its mining algorithms are sequential algorithms. Existing parallel frequent pattern mining algorithms can be extended to mine correlated patterns. However, such a naive parallel algorithm is inadequate to discover correlated patterns effectively. It is because naive parallel algorithm cannot exploit the properties of all-confidence measure to discover the patterns effectively. In this paper, we have proposed an efficient parallel correlated pattern mining algorithm by introducing two novel data segmentation techniques. Experimental results have demonstrated that proposed algorithm is efficient.

In this paper, we have extended the proposed data segmentation techniques to Map-Reduce framework. As a part of future work, we would like extend the proposed techniques to other distributed frameworks, such as cluster and grid networks.

Acknowledgement. This research was partly supported by Real World Information Analytics project of National Institute of Information and Communications Technology, Japan.

References

1. Agarwal, R., Srikant, R., et al.: Fast algorithms for mining association rules. In: Proceedings of the 20th VLDB Conference, pp. 487–499 (1994)
2. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: ACM SIGMOD Record, vol. 29, no. 2, pp. 1–12. ACM (2000)
3. Brin, S., Motwani, R., Silverstein, C.: Beyond market baskets: generalizing association rules to correlations. In: ACM SIGMOD Record, vol. 26, no. 2, pp. 265–276. ACM (1997)
4. Omiecinski, E.R.: Alternative interest measures for mining associations in databases. *IEEE Trans. Knowl. Data Eng.* **15**(1), 57–69 (2003)
5. Xiong, H., Tan, P.-N., Kumar, V.: Hyperclique pattern discovery. *Data Min. Knowl. Discov.* **13**(2), 219–242 (2006)
6. Yun, H., Ha, D., Hwang, B., Ryu, K.H.: Mining association rules on significant rare data using relative support. *J. Syst. Softw.* **67**(3), 181–191 (2003)
7. Tan, P.-N., Kumar, V., Srivastava, J.: Selecting the right interestingness measure for association patterns. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 32–41. ACM (2002)
8. Wu, T., Chen, Y., Han, J.: Re-examination of interestingness measures in pattern mining: a unified framework. *Data Min. Knowl. Discov.* **21**(3), 371–397 (2010)
9. Lee, Y.-K., Kim, W.-Y., Cai, Y.D., Han, J.: Comine: efficient mining of correlated patterns. In: ICDM, vol. 3, pp. 581–584 (2003)
10. Kim, W.-Y., Lee, Y.-K., Han, J.: CCMine: efficient mining of confidence-closed correlated patterns. In: Dai, H., Srikant, R., Zhang, C. (eds.) PAKDD 2004. LNCS (LNAI), vol. 3056, pp. 569–579. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24775-3_68
11. Kim, S., Barsky, M., Han, J.: Efficient mining of top correlated patterns based on null-invariant measures. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011. LNCS (LNAI), vol. 6912, pp. 177–192. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23783-6_12
12. Uday Kiran, R., Kitsuregawa, M.: Efficient discovery of correlated patterns in transactional databases using items' support intervals. In: Liddle, S.W., Schewe, K.-D., Tjoa, A.M., Zhou, X. (eds.) DEXA 2012. LNCS, vol. 7446, pp. 234–248. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32600-4_18
13. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. *Commun. ACM* **51**(1), 107–113 (2008)
14. Li, H., Wang, Y., Zhang, D., Zhang, M., Chang, E.Y.: PFP: parallel FP-growth for query recommendation. In: Proceedings of the 2008 ACM conference on Recommender systems, pp. 107–114. ACM (2008)
15. Xun, Y., Zhang, J., Qin, X.: Fidoop: parallel mining of frequent itemsets using mapreduce. *IEEE Trans. Syst. Man Cybern. Syst.* **46**(3), 313–325 (2016)
16. Pramudiono, I., Kitsuregawa, M.: Parallel FP-growth on PC cluster. In: Whang, K.-Y., Jeon, J., Shim, K., Srivastava, J. (eds.) PAKDD 2003. LNCS (LNAI), vol. 2637, pp. 467–473. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36175-8_47
17. Chen, D., Sain, S.L., Guo, K.: Data mining for the online retail industry: a case study of RFM model-based customer segmentation using data mining. *J. Database Mark. Cust. Strateg. Manag.* **19**(3), 197–208 (2012)



Time Series Distance Density Cluster with Statistical Preprocessing

Ruizhe Ma^(✉), Soukaina Filali Boubrahimi, and Rafal Angryk

Georgia State University, Atlanta, GA 30303, USA
{rma1,sfilaliboubrahimi1}@student.gsu.edu, rangryk@cs.gsu.edu

Abstract. Previous Distance Density Clustering has shown some promising results for univariate time series datasets. However, due to the nature of time series data and from using Dynamic Time Warping algorithm as the distance measure; Distance Density Clustering is not an efficient heuristic with larger datasets. In this paper we propose a preprocessing step that could augment the algorithm to the parallel case, and speed up the Distance Density Clustering process considerably. We use time series sequence feature: peak numbers, to prune impossible matchings. By doing so, we are able to form preliminary feature clusters, and further clustering is applied within each feature cluster individually. This can narrow down the amount of time series distance computations, and make Distance Density Clustering scalable.

Keywords: Time series · Clustering

1 Introduction

Clustering is an important branch of exploratory data mining. The goal of which is to minimize intra-cluster distance, and maximize inter-cluster distance. With wide application in pattern recognition, machine learning, image analysis, and bioinformatics; discrete clustering problems are very well studied. However, with widespread availability and usage of time series data, many existing clustering approaches are found to be lacking in performance. Clustering time series data, even in the univariate case can be a challenging task. Main issues include distance measure and cluster representation, as well as the time cost of processing time series data.

Many models exist for clustering non time series data. In fact, the notion of cluster cannot be precisely defined, which is the reason of so many existing clustering algorithms [6]. Clustering can be categorized as *Hard Clustering*, where each object either belongs to a cluster or not; and *Soft Clustering*, where an object belongs to each cluster to a certain degree. Clustering can also be categorized based on the specific methodology. Some commonly used clustering algorithms include *hierarchical clustering*, which is based on distance connectivity. Hierarchical clustering is greedy, with no look back adjustments, but easy to

combine with any distance measure. *K-means* algorithm [8] is based on the distance between element/events, it is not a definitive algorithm, and initialization can heavily influence the end results. The number of clusters also needs to be determined *a priori* for k-means, which can make it difficult for exploratory analysis. *DBSCAN* (Density Based Spatial Clustering of Application with Noise) [7] is based on density, it is less efficient than the previous two methods, but does not require the user to predetermine the number of clusters. *DBSCAN* is also known to be more effective with arbitrary shaped clusters compared to k-means, and more robust to outliers. However, *DBSCAN* cannot detect clusters with different densities.

Some preliminary works on time series clustering involves feature extraction such as mean, median, variance, peak number, etc. This works to an extent, however, this contradicts the original aim of using time series, which is the attention to details and trends. We can augment existing clustering methods to be used for time series, by combining elastic measures with existing cluster algorithms. For example, by combining DTW (Dynamic Time Warping) with hierarchical clustering, time series data can be clustered by dendrograms [13]. Additionally, DBA (DTW Barycenter Averaging) [3] can be introduced to extend k-means to time series usage. The DDC (Distance Density Clustering) [2] algorithm combines DTW, DBA and the concept of time series virtual density for a deterministic time series clustering.

When a dataset is within reasonable size for human analysis, human perception of similarity can be very efficient. Which is reason domain experts were heavily relied on for drawing conclusions from collected data in the past. Nowadays however, the amount and speed at which data are collected, makes it impossible for human to process. Yet, we can take intuition of how we perceive similarity. When considering various sequences, we are able to notice the significant features such as the number and shape of peaks. After a preliminary evaluation, we can zoom in to the details and actual values to further differentiate between sequences. Inspired by this, we want to introduce a preliminary evaluation of time series data by the computer before moving in to the more time consuming time series clustering.

This paper aims to address the problem of slow time series clustering. Our objective is to apply a simple, low-cost statistical feature preprocessing step to eliminate impossible groupings in time series clustering. Our goal is to parallel the clustering process, improve computation efficiency, while maintaining comparable accuracy. This can boost the speed of the otherwise time expensive DDC algorithm as well as other similar algorithms. The rest of this paper is organized as follows: Sect. 2 introduce the background information involved, including distance measure, cluster representation, and time series clustering. Section 3 discuss how smoothing and feature selection is used to improve Distance Density Clustering. Section 4 show the experimental results in terms of speed and accuracy. Finally, Sect. 5 concludes this paper.

2 Background

2.1 Distance Measure

Distance measure can be generalized as lock-step measures and elastic measures. Distance measure is very important in determining the similarity between data objects, which further helps to assign data objects to different groups or clusters. Euclidean distance is a classic lock-step distance measure that is widely used in the past as the “standard” distance measure. Euclidean distance is the “ordinary” distance between two points in the Euclidean space, Eq. 1. However, with the advent of time series data, Euclidean distance provides less satisfactory results. Given two time series sequences Q and C , $Q = \{q_1, q_2, \dots, q_i, \dots, q_n\}$, and $C = \{c_1, c_2, \dots, c_j, \dots, c_m\}$, when measuring the distance between time series data, lock-step measures compare the i th element of time series Q to the i th element of time series C . Making lock-step measures sensitive to noise and misalignments in time [9].

A popular elastic measure is the Dynamic Time Warping (DTW) algorithm, it is used for measuring the similarity between two temporal sequences which may vary in time or speed. Originally, DTW was used in speech recognition, later it was adapted to various real-world data mining problems. This method allows computers to find a relatively optimal match between two given sequences under certain restrictions. Its advantage comes from the allowance of one-to-many mappings. Due to the stretching and compressing allowed in DTW, the compared sequences is not required to be of uniform length. Equation 2 is the distance function for DTW. When calculating the DTW distance, an n -by- m distance matrix is first constructed, which contains the distance information between all the elements from the two sequences, often using Euclidean distance. The DTW algorithm then seeks a relatively shorter path through the distance matrix. The warping path is denoted as $W = \{w_1, w_2, \dots, w_k, \dots, w_K\}$, only the path minimizing $Dist(DTW)$ is used.

$$Dist(Euclidean) = \sqrt{\sum_{i=1}^n (q_i - c_i)^2} \quad (1)$$

$$Dist(DTW) = \min\left\{\sqrt{\sum_{k=1}^K w_k/K}\right\} \quad (2)$$

The elastic DTW mapping between time series is often times more intuitive and close to human perception than the Euclidean mapping in. DTW accounts for the similar shapes even when they are not a direct temporal match. The effectiveness of DTW has been extensively explored in various literatures [10–13]. Hence, we will not further compare the performance advantage.

2.2 Cluster Representation

The difficulty of clustering time series data comes largely from a lack of cluster representation. Local averaging methods are sensitive to computational order,

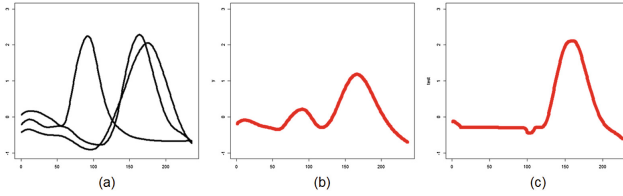


Fig. 1. (a) Three out-of-phase but similar time series of the same class, (b) the average of the three time series in (a) using traditional averaging, (c) the average of the three time series in (a) using DBA, with the middle peak sequence as template.

and can cause the average sequence to increase in length. Traditional averaging using Euclidean distance typically generate sequences that are not similar to the time series it is representing. Petitjean et al. proposed a global method to calculate the “average” of time series data, the DTW Barycenter Averaging (DBA) [3]. Barycenter refers to the orbiting mass center. For time series data, we can assume each time series sequence as a mass, and the average sequence is the orbiting center. And this orbiting center is refined with DTW between time series and initial average by minimizing the Within Group Sum of Squares (WGSS). Given a time series set $\mathbb{S} = \{S_1, S_2, \dots, S_n\}$, the time series $C = \{c_1, c_2, \dots, c_t\}$ is considered an average of \mathbb{S} if it minimizes $WGSS(C) = \sum_{k=1}^n dtw(C, S_k)^2$

DBA is a global time series averaging method, meaning it has no sensitivity to the order in which the average sequence is computed. Figure 1 shows the traditional averaging and DBA results when given three similar but out-of-phase sequences. The traditional averaging method of dividing the sum value at each temporal point provides unintuitive results with time series data. Shown in Fig. 1(b), traditional averaging generated two flatter peaks, this is obviously a poor representation of the three sequences. Whereas the DBA result is more intuitive. DBA is sensitive to the initial averaging template. When there are few sequences being averaged, the DBA often times bear more resemblance to the initial template. Thus the reason Fig. 1(c) is most similar to the second peak sequence in Fig. 1(a).

2.3 Time Series Clustering

The definition of clustering is imprecise, and depending on definition, the problem can be approached from many perspectives. In our previous study, we proposed the Distance Density Clustering (DDC), a deterministic clustering method [2]. DDC is the combination of a centroid model and a density model, meaning it takes both distance and density into consideration during clustering, and it has shown some promising results [14].

The DDC algorithm computes a DTW distance matrix. From a general voting process, the time series sequence that has the furthest distance to the most number of sequences is chosen as the initial cluster seed, we refer to this as the furthest seed. Then we order the distance from all other sequences to the furthest

seed in an ascending manner. The largest increase in the distance is a virtual sparse region of time series. Virtual, because time series is not the same as data that can be plotted on a plane coordinate system. Therefore, dense and sparse regions cannot be accurately defined. Instead, we mimic density and sparsity with the DTW distance between time series. Because each split will result in two clusters, the number of clusters will increase exponentially. In order to avoid this, each split after the initial one chooses the steeper increase from the previous two clusters to split. Clusters are rebalanced based on the distance to the seeds, which are the medoids to the corresponding DBAs.

In DDC, each cluster is represented with the medoid to the DBA of the cluster. The most representative time series of a group of time series is more likely to be from a virtual dense region of the group. Therefore, the DBA average template is determined by a majority voting strategy, where the event that is the nearest event to the most time series is chosen. As such, the entire DDC process is deterministic and reproducible. For more details of DDC and examples, please refer to Ma et al. [2].

3 Statistical Preprocessed DDC



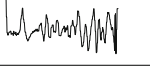
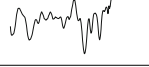
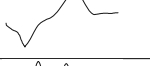
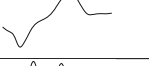


3.1 Feature Selection

In this paper we use peak numbers to pre-process the data into large clusters to be further clustered by DDC. However, with many time series data, the sequences can be too noisy to find meaningful peaks. Raw time series data is not an ideal source to obtain features such as peaks. As such, smoothing needs to be done to help eliminate small peaks before actual peak identification. We will refer to the large clusters obtained after smoothing and feature selection as *feature cluster*. Here we are more interested in efficiency than precision, as it only serves us to gain a rough understanding of the data overall shape, and further processing will take place on the original time series. Therefore we use the simple moving average to obtain the overall peak feature. This way, the peak numbers can create a range of possible comparison for time series, and impossible comparisons are eliminated, thus saving time.

Due to the noisy nature of some real-world time series data, even after smoothing, we do not wish to select every peak in the time series sequences. Instead we wish to find the “significant” peaks to differentiate time series. A peak can be naïvely defined as satisfying $c_i > c_{i-1}$ and $c_i > c_{i+1}$, when temporal index i corresponds to the peak c_i . Here we refer to every peak that satisfies this description as a *candidate peak*. Parameters used for peak identification are as follows:

- t : the threshold on the frequency domain, insignificant peaks below the threshold can be filtered.
- d : the minimum peak radius distance. For any significant peak, its adjacent insignificant (i.e., relatively short) peaks can be seen as noise and therefore ignored.

Table 1. Feature cluster example

Event	Original Sequence	Smoothed Sequence	Peak Number (p)
a			0
b			7
c			2
d			3

- n : the maximum number of peaks to be taken into account. Since the peaks can be sorted with respect to their significance (i.e., their height in the time series), this parameter can push a constraint on the number of significant peaks.

All candidate peaks are first identified and sorted based on their value. Then, a filtering process is applied, first, the threshold t on the frequency domain is applied, then for each of the remaining peaks, its neighboring peaks within the radius of d are removed. Here the peaks are processed in a descending order, maintaining the survival of the local maximum peaks. Among the remaining peaks, the first n peaks are returned. This process ensures only peaks that are significant enough are used to identify and differentiate time series sequences. Since peaks are the significant features, we are more selective with the identification parameter selection. We can fine tune peak identification parameters for a single dataset, however, this would not scale well to other datasets. For a more general choice, we used the third quantile as threshold t , and one tenth of sequence length as peak identification radius d .

In Table 1, four time series a , b , c , and d are shown with the original sequence and smoothed sequence, as well as the number of identified peaks p . Due to small irregularities, sometimes similar sequence are identified with slightly different number of peaks. So we cannot simply place same peak number sequences together. For each dataset, the number of identified peaks is ordered, and starting from the smallest p , all sequences with peak number $\pm 2p$ are placed within the same feature cluster. In the example shown in Table 1, time series a has 0 peaks, and is therefore in its own feature cluster. The next smallest p is time series c with 2 peaks, which means it is grouped together with time series d , which has 3 peaks. Finally, time series b has its own feature cluster. By taking out time series out of the candidate list, we are able to avoid duplicate time series appearing in different feature clusters.

3.2 Distance Density Clustering with Statistical Preprocessing

With smoothing and feature selection as preprocessing steps to narrow the clustering possibilities, we conduct DDC within each feature cluster. The first step is to compute the distance matrix for each feature cluster. This means that the distance information between events from different feature clusters will not be calculated. When using elastic measures for very large datasets, this alone can save a considerable amount of time. Additionally, this also divides a dataset into sections, meaning the clustering of each feature cluster can be processed in parallel.

Typically, the smaller the cluster, the higher the testing classification accuracy. On the other hand, smaller cluster usually means longer training time. This has proven to be an issue in the past, especially with large datasets [2]. Interestingly, because feature cluster can augment the traditional DDC to become a parallel heuristic, the more feature clusters we obtain the faster the clustering process. However, increasing cluster numbers need to be done with care, as rare occurring events can be overshadowed when a cluster is labeled with the majority label of that cluster.

Algorithm 1 in Appendix A shows how feature preprocessing is used in conjunction with the DDC. Every time series sequence is first smoothed. Then the significant peaks are identified for each sequence. Each sequence is placed in feature clusters with sequences of similar number of peaks. And DDC is carried out within each feature cluster in parallel. We then identify the largest increase of accuracy when the number of clusters is increased by one, this determines the number of clusters in each feature cluster. In line 17–19, each cluster is labeled by the majority event label, and the DBA of each cluster is then computed. During testing, line 21–26, each time series is compared to the DBA of each cluster and classified. The testing label is then returned and compared to the benchmark labels.

4 Experiments

In this section we show how DDC with statistical preprocessing of peak features compares with the original DDC. We compare the training time and testing accuracy between the two approaches. For our experiments, we selected 20 datasets with varying lengths and sizes from a public repository, the UCR time series dataset archive [4]. When processing each dataset using the DDC method, events are clustered in a divisive manner, which is hard to process in parallel. In feature DDC, the divisive clustering takes place within each feature cluster of each dataset, which is easy to process in parallel. The amount of parallelization depends on the number of identified feature clusters, in our experiments the upper limit is 50 cores.

Figure 2(a) and (b) share the same horizontal axis, labeled with datasets. Figure 2(a) shows the training time comparison between DDC and feature DDC, with time on the vertical axis. The processing time are very contrasting between different datasets, therefore, the time is in logarithmic scale. The feature DDC

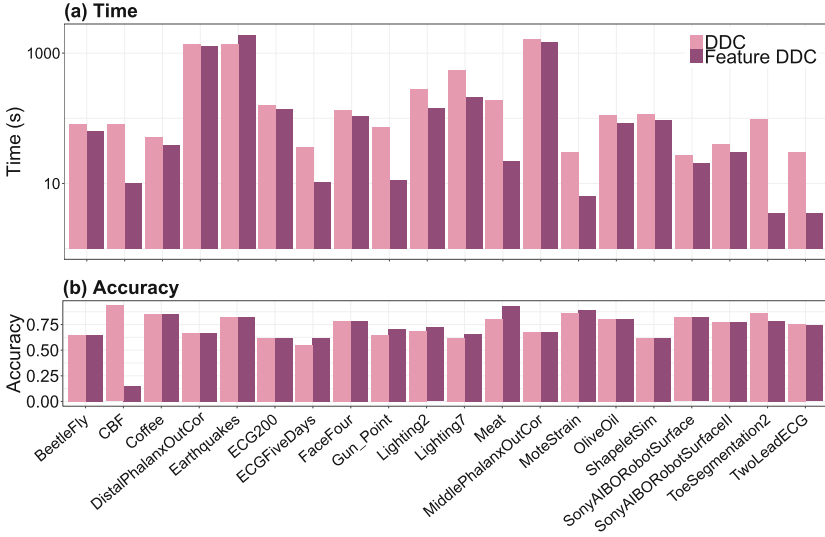


Fig. 2. Training time of DDC compared to feature DDC

training time includes the time spent on peak feature selection. Theoretically, because it can cut a dataset into many feature clusters, feature DDC should speed up DDC. However, that is not always the case. Feature DDC has the best performance when there is a distinct peak difference between at least some dataset classes. The best scenario for feature DDC would be clear feature distinction between classes, as well as a large number of classes. When peak number cannot sufficiently separate the dataset, feature DDC degenerates to the standard DDC. Out of the 20 datasets, there are two cases where feature DDC spent slightly more time.

Figure 2(b) compares the accuracy achieved by DDC and feature DDC, with accuracy on the vertical axis. Our intent is to speed up DDC while maintaining comparable accuracy, and overall the accuracies between the two approaches are similar. With 17 out of the 20 cases where feature DDC has equal or better accuracy. For datasets “CBF”, “ToeSegmentation2”, and “TwoLeadECG”, feature DDC has lower accuracy than the standard DDC, especially for dataset “CBF”. These 3 datasets share a similar characteristic: the ratio of training to testing data is very small. This happens because not all event labels are occurring equal number of times in the datasets. When at least one class of rare occurring events get separated during feature clustering, they could be masked and clustered into a cluster where the majority has a different label. This would cause a rare label to disappear during training process, causing low testing accuracy. As is the case for dataset “CBF”.

The performance of clustering algorithm is often times dataset dependent. It is difficult to develop a one-algorithm-fits-all heuristic. The main goal of this paper is to ameliorate the high time consumption of the standard DDC approach,

and extend it to the parallel case. As we have shown through experiments, in most cases DDC with statistical peak preprocessing has shorter training time, while maintaining comparable accuracy.

5 Conclusion

In this paper we introduced a statistical preprocessing step used to speed up Distance Density Clustering, a clustering algorithm for time series data. By using the number of peaks to pre-cluster the data into feature clusters we are able to augment DDC to the parallel case. Which is especially important for large time series datasets. The drawback of this approach is its data dependency, as shown in the experiments. For datasets with extremely similar data between different classes, or rare occurring class events, statistical preprocessing could become risky.

In the future, we plan to use a combination of statistical features to pre-cluster larger and more complex time series dataset. For larger datasets, the heuristic will have to be altered to best suit the single dataset on hand. This will make the algorithm more efficient, and should also provide better accuracy performance.

Acknowledgment. This project has been supported in part by funding from the Division of Advanced Cyber infrastructure within the Directorate for Computer and Information Science and Engineering, the Division of Astronomical Sciences within the Directorate for Mathematical and Physical Sciences, and the Division of Atmospheric and Geospace Sciences within the Directorate for Geosciences, under NSF award #1443061. It was also supported in part by funding from the Heliophysics Living With a Star Science Program, under NASA award #NNX15AF39G.

A Appendix

Algorithm 1. Feature Preprocessed Distance Density Clustering Algorithm

Require:

DistMtrx distance matrix of all time series events
TrainDataSet = {*train*₁, *train*₂, ..., *train*_{*n*}}
TestDataSet = {*test*₁, *test*₂, ..., *test*_{*m*}}
*cluster*_{*i*} = {*clusterevents*}
TrainLabel = {*label*_{*train*₁}, *label*_{*train*₂}, ..., *label*_{*train*_{*n*}}}
1: Smooth *TrainDataSet* time series
2: Find peaks array *peaksArray* for each event in the smoothed *TrainDataSet*
3: *peaksArray* = {*peaksNum*₁, *peaksNum*₂, ..., *peaksNum*_{*n*}}
4: *clusterEvents* = *PeaksCluster*(*peaksArray*)
5: *clusterEventsProcessed* = *ProcessClusterList*(*clusterEvents*)
6: **for** *cluster*_{*i*} ∈ *clusterEventsProcessed* **do**
7: *DistMtrx*_{*sub*} = *DistMtrx*[*cluster*_{*i*}, *cluster*_{*i*}]
8: **if** *length*(*cluster*_{*i*}) < *threshold* **then**
9: *clusterDBA*_{*i*} = *DBA*(*DistMtrx*_{*sub*})
10: *clusterElements*_{*i*} = *cluster*_{*i*}
11: **else**
12: (*clusterElementsList*, *clusterDBAList*) = *DDC*(*DistMtrx*_{*sub*})
13: **end if**
14: **end for**
15: *clusterElementsList* = {*clusterElements*₁, *clusterElements*₂, ..., *clusterElements*_{*p*}}
16: *clusterDBAList* = {*clusterDBA*₁, *clusterDBA*₂, ..., *clusterDBA*_{*p*}}
17: **for** *clusterElements*_{*i*} ∈ *clusterElementsList* **do**
18: *clusterTrainLabel*_{*i*} Cluster labeled with majority event label
19: **end for**
20: *clusterTrainLabelList* = {*clusterTrainLabel*₁, *clusterTrainLabel*₂, ..., *clusterTrainLabel*_{*p*}}
21: **for** *testData*_{*i*} ∈ *TestDataSet* **do**
22: **for** *clusterDBA*_{*i*} ∈ *clusterDBAList* **do**
23: *dtwDistance*_{*i*} = *dtw*(*testData*_{*i*}, *clusterDBA*_{*i*})
24: **end for**
25: *label*_{*test*_{*i*}} = *clusterTrainLabel*_{*minDist*}
26: **end for**
27: **return test classification results** *TestLabel* = {*label*_{*test*₁}, *label*_{*test*₂}, ..., *label*_{*test*_{*m*}}}

References

1. Savitzky, A., Golay, M.J.E.: Smoothing and differentiation of data by simplified least squares procedures. *Anal. Chem.* **36**(8), 1627–1639 (1964)
2. Ma, R., Angryk, R.: Distance and density clustering for time series data. In: 2017 IEEE International Conference on Data Mining Workshops (ICDMW). IEEE (2017)
3. Petitjean, F., Ketterlin, A., Gançarski, P.: A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recogn.* **44**(3), 678–693 (2011)
4. Chen, Y., Keogh, E., Hu, B., et al.: The UCR time series classification archive (2015). www.cs.ucr.edu/~eamonn/time_series_data
5. Bagnall, A., et al.: The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining Knowl. Discov.* **31**(3), 606–660 (2017)

6. Estivill-Castro, V.: Why so many clustering algorithms: a position paper. *ACM SIGKDD Explor. Newslett.* **4**(1), 65–75 (2002)
7. Ester, M., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: *KDD*, vol. 96, no. 34 (1996)
8. Lloyd, S.: Least squares quantization in PCM. *IEEE Trans. Inf. Theory* **28**(2), 129–137 (1982)
9. Ranacher, P., Tzavella, K.: How to compare movement? a review of physical movement similarity measures in geographic information science and beyond. *Cartography Geog. Inf. Sci.* **41**(3), 286–307 (2014)
10. Berndt, D.J., Clifford, J.: Using dynamic time warping to find patterns in time series. In: *KDD Workshop*, vol. 10(16), pp. 359–370 (1994)
11. Keogh, E., Ratanamahatana, C.A.: Exact indexing of dynamic time warping. *Knowl. Inf. Syst.* **7**(3), 358–386 (2005)
12. Rakthanmanon, T., Campana, B., Mueen, A., et al.: Searching and mining trillions of time series subsequences under dynamic time warping. In: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 262–270. ACM (2012)
13. Ma, R., Angryk, R., Riley, P.: A data-driven analysis of interplanetary coronal mass ejecta and magnetic flux ropes. In: *2016 IEEE International Conference on Big Data (Big Data)*. IEEE (2016)
14. Ma, R., et al.: Coronal mass ejection data clustering and visualization of decision trees. *Astrophys. J. Suppl. Ser.* 236(1), P. 14 (2018).



Debate Stance Classification Using Word Embeddings

Anand Konjengbam^(✉), Subrata Ghosh, Nagendra Kumar, and Manish Singh

Indian Institute of Technology Hyderabad, Sangareddy 502285, India
{cs14resch11004,cs15mtech11014,cs14resch11005,msingh}@iith.ac.in

Abstract. Online debate sites act as a popular platform for users to express and form opinions. In this paper, we propose a novel unsupervised approach to perform stance classification of two-sided online debate posts. We propose the use of word embeddings to address the problem of identifying the preferred target of each aspect. We also use word embeddings to train a supervised classifier for selecting only target related aspects. The aspect-target preference information is used to model the stance classification task as an integer linear programming problem. The classifier gives an average aspect classification accuracy of 84% on multiple datasets. Our word embedding based stance classification approach gives 19.80% higher user stance classification accuracy (F1-score) compared to the existing methods. Our results suggest that the use of word embeddings improves accuracy and enables us to perform stance classification without the need for external domain-specific information.

Keywords: Two-sided online debate · Stance classification
Text mining

1 Introduction

In this era of Internet technology, online debates are becoming more and more popular. Many users get involved in a variety of online debates such as forums, blogs, social media sites, etc. These debates contain rich information that can be exploited in various services or decision making processes. Especially, understanding the users' opinion and their stance on these debates can provide valuable information in the decision making process of governments, companies, and can help shape public opinion towards issues.

Stance detection is the process of discovering the standpoint of users in an online debate post. A debate consists of a topic of discussion, targets, and posts from various authors. A target may be a product, technical topic, political side, ideological belief, social issue, etc. For example, consider the debate post: “*Windows supports most of the games. Mac has a lot of software compatibility issue*”. Here, the topic of discussion is “*Is Mac better than Windows?*”, and the targets are *Windows* and *Mac*. One can deduce that the author is writing favourably about *Windows* and drawbacks of *Mac*.

Stance detection incorporates a new dimension to other related areas like information retrieval, discourse analysis, fake news detection [13], and recommendation system. For example, by applying stance detection on a product debate, we can find out the product preference of users. Advertisement of related products can be displayed to users according to their stance. In this paper, we study two-sided debate posts, where there are two opposing targets. The user may write about the good points of her preferred target or bad points about why she is against the other.

There are two main approaches for stance detection, namely, supervised and unsupervised. Most of the existing stance detection methods follow supervised approach [1, 5]. These methods are not scalable as for each domain, they require a large data set to be collected and annotated for training stance classifiers. The advantage of unsupervised approaches is that they do not require any manual data labeling [3, 9, 14]. These unsupervised methods can easily adapt to a variety of domains.

Unsupervised methods typically use linguistic and syntactic dependency relations between targets and opinions. Most of the existing unsupervised methods focus on stance detection on tweets [5, 11]. Since tweets are very short (140 characters or less) and they contain useful target and opinion information in the form of hashtags and emoticons, it is relatively easy to identify targets and opinions compared to online debate posts, which are typically long and do not have markers such as hashtags. In this paper, we focus on unsupervised stance detection on online debate posts, which are on average 600 characters long and have no tags. We extend the work of Somasundaran and Weibe [14], which performs unsupervised stance detection on long debate posts and does not require any marker information.

The main challenge in stance detection is to identify the preferred target of each aspect. The sentiment of the text or aspects alone is not enough to determine the stance as debate posts do not contain sufficient information to determine the aspect-target preference. Sometimes, a text may not contain any target and opinions may be expressed towards unrelated targets. To better understand the problem, consider the following example of a debate post:

Example 1. “Everyone knows that Windows is the better operating system because it has more softwares, more developer support, more everything. Also, Windows gives more freedom to users. Mac is like the retarded little brother who is too much pampered. It has Wi-Fi connection problem and lacks gaming support.”

Example 1 is of a debate post on the topic *Windows vs. Mac*. Here, the user writes about why she is supporting one target (*Windows*) over the other (*Mac*). Since the user writes about both the targets, it is hard to figure out which target each aspect (e.g., *operating system*, *software*, and *gaming*) is supporting. Also, the aspect *brother* is not related to the debate targets. We need to rely on some external sources, such as web corpus to extract aspect-target relation. Existing approaches need to crawl web pages related to a given debate topic to get the aspect-target relationship [14]. These approaches are computationally expensive

and time consuming. Besides, the accuracy of such approach is limited by the availability of domain information from web corpus. Even using web corpus, we may have access to limited number of pages, many of which may not be even relevant.

In this paper, we propose the use of word embeddings to do unsupervised stance detection. Word embedding is a neural-network based language modelling method used to represent text as dense vectors. It can encode precise semantic and syntactic relationships between words and can also provide useful information about co-occurrence of words [10]. We utilize this property of word embeddings to find aspects that are relevant to the debate targets. We also use it to find aspect-target preference.

The remaining paper is organized as follows: Sect. 2 outlines the previous related works. We discuss our proposed approach in Sect. 3. Section 4 details the evaluations. We finally conclude the paper in Sect. 5.

2 Related Works

Existing work on debate stance classification can be broadly categorized into two settings, viz. (1) congressional floor debates, and (2) social debates. The congressional floor debates are the more professional debates, whereas the social debates are informal. Stance classification in social debates is more challenging compared to congressional floor debates due to the expressive language, such as slangs, lack of grammar, abbreviations, etc., used in the social discussions [16]. Stance classification of congressional floor debates investigates political orientation of users on issues such as political reforms, controversial policies, and candidate preference [2, 15]. Stance detection of social debates aim at detecting the users' stance towards topics in informal settings such as online public forums and debate websites¹ [2, 14]. In this paper, we perform stance detection for social debates, where the post length is long compared to tweets.

As discussed in the introduction, our work is an extension to the work of Somasundaran et al. [14]. We employ word embedding techniques in various steps of stance detection. This approach speeds up stance classification process and gives more accurate result compared to the earlier approach that requires crawling of external information related to the targets to find aspect-target preference information. Researchers have used word embeddings for opinion mining due to its ability to capture semantic similarities [8, 17]. Liu et al. [8] used word vectors trained from a large corpus of product reviews for opinion aspect extraction. Xu et al. [17] used word embeddings as feature representation for a joint opinion relation detection. Mikolov et al. [10] developed *word2vec* model for creating neural-embedded word representations. Levy and Goldbery [6] have shown that *word2vec* model captures relational and semantic similarities better than other methods.

¹ <http://www.convinceme.net>, <http://www.createdebate.com>, and <http://www.debate.org> are some of the popular social debate sites.

Our work is also related to the aspect extraction step of opinion mining. A lot of work has been done in the area of aspect extraction for opinion mining [4, 7]. Frequent nouns and noun phrases extracted from user reviews are used as aspects in [4]. Liu et al. [7] proposed a language pattern mining method to extract aspects from pros and cons reviews. Recent works have focused on using word embeddings for aspect extraction [8]. Identifying the target related aspects from a debate post is a more challenging task as there are more than one targets involved in a debate post. Somasundaran and Wiebe [14] used syntactic rules to extract aspect from debate posts. However, many of the extracted aspects are irrelevant to the debate targets. In this paper, we proposed a novel word embedding based method to train a supervised aspect classifier to prune aspects irrelevant to the targets. The classifier can be used for transfer learning to find target related aspects across multiple domains.

3 Proposed Approach

In this section, we describe the general architecture of the proposed Debate Stance Classification using Word Embeddings (DSCW), and then, highlight the specific areas of the problem that we aim to solve.

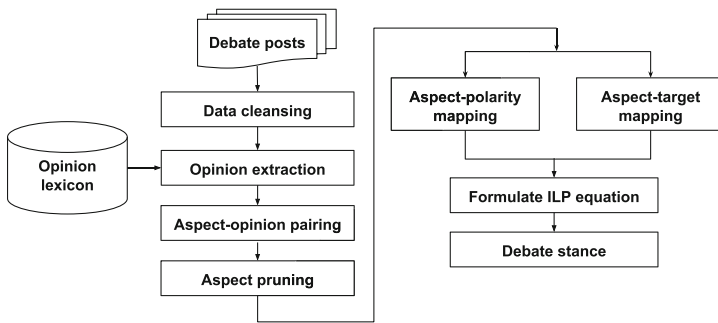


Fig. 1. Workflow of debate stance classification using word embeddings.

The architectural overview of the proposed system is shown in Fig. 1. It takes a two-sided debate post as input. The output is the standpoint of the user in the debate post. It performs the following steps to generate the output: (1) data cleaning, (2) extracting opinion words, (3) creating aspect-opinion pairs, (4) pruning irrelevant aspects using word-embeddings, (5) creating aspect-polarity pairs, (6) creating aspect-target preference using word-embeddings, and (7) detecting debate post stance using Integer Linear Programming (ILP). As it can be seen in Fig. 1, unlike the work proposed by Somasundaran and Wiebe [14], our focus is to use word embeddings for finding aspect-target preference instead of creating a probability preference table, which requires external web corpus. We also introduce three additional steps namely, data cleaning,

lexicon expansion, and aspect pruning, which differentiate our proposed approach from the existing one. We next explain these steps in detail.

3.1 Data Cleaning

Users often do not follow proper grammatical rules and use short forms while writing online social debate posts. The syntactic rules used in unsupervised methods will not perform well if the sentences are not grammatically correct. Therefore, we preprocess the debate posts to make it grammatically correct, and also clean noisy and inconsistent data. For preprocessing, we perform stop word removal (e.g., *what* and *an*), removal of repeated character sequence (e.g., *besttttt*), URL filtering, acronym expansion, contraction substitution, emoticon substitution (e.g., “:”) $\rightarrow +1$; “:(” $\rightarrow -1$), and lemmatization.

3.2 Opinion Word Extraction

After cleaning and refining the post, the next step is to extract opinion words from the cleaned post. To address this issue, researchers turn to opinion lexicons [4, 14], which is a compilation of opinion words along with the associated semantic polarity (e.g. *good* : +1; *bad* : -1; *basically* : 0). If the opinion lexicon is small, we could lose out opinion words present in debate posts, which are in turn, used to detect aspects and find the stance of the post. In order to increase the coverage of opinion words across multiple domains, we combine multiple publicly available opinion lexicons [4, 12, 14]. In addition, we also identify comparative adjectives (e.g., *better*) and superlative adjectives (e.g., *best*) from posts using Stanford Part Of Speech (POS) tagger² and add them to our opinion lexicon (O_L). These comparative words appear frequently in debate posts and express sentiments. The polarity associated with these words are obtained using SentiWordNet³. After merging all these lexicons, the size of our O_L is almost 1.7 times that of the opinion lexicon size used in [14].

3.3 Aspect-Opinion Pairing

This step identifies the aspects associated with opinion words extracted from debate posts. We use the unsupervised rule-based method proposed by Somasundaran and Wiebe [14] to extract aspect-opinion pairs (AOP). They leveraged the property that opinions are expressed on aspects of the targets. Their method identifies dependency relations between aspects and opinions using syntactic dependency trees generated by the Stanford parser⁴. Based on the dependency relations, they proposed five syntactic rules (DIRECT OBJECT rule, NOMINAL SUBJECT rule, ADJECTIVAL MODIFIER rule, PREPOSITIONAL OBJECT rule, and RECURSIVE MODIFIER rule) to pair opinions with aspects.

² <https://nlp.stanford.edu/software/tagger.shtml> .

³ <http://sentiwordnet.isti.cnr.it/>.

⁴ <https://nlp.stanford.edu/software/lex-parser.shtml>.

3.4 Aspect Pruning

We observe that more than 80% of the aspects extracted using the syntactic rules are not related to the debate topics. For instance, in Example 1, the sentence ‘*Windows gives more freedom to users*’, has the aspect *freedom*, which is neither related to *Windows* nor *Mac*. Similarly, when we manually inspect the aspects extracted from the ‘*Firefox Vs Internet Explorer (IE)*’ debate post, we observe that out of 552 aspects obtained using the syntactic rules, only 113 are found to be relevant to either of the debate topics. We have identified five types of irrelevant aspects that need to be pruned out as such aspects do not contribute in deciding the stance of the debate post. They are described in Table 1.

Table 1. Classes of irrelevant aspects obtained after applying the syntactic rules.

Class	Sample irrelevant aspects	Example debate posts
Pronouns	anyone, everyone, everything, something, who, which, us, they	Why does Microsoft continue developing internet explorer if everyone thinks of this browser as a joke compared to other browsers
Proper nouns	ALT, AOL, W3C, Apple, IIS, IM, MS, Vista, Warner, Norton	Norton blocks off IE from any updates as IE is filled with many ads and pop-ups
Common nouns in related sentences	approach, area, argument, article, base, beholder, example	IE’s bloated approach to software have frequently prevented it from rolling out much sought after changes and improvements
Common nouns in unrelated sentences	cars, bank, availability, battle, beast, brother, business, case	Just because IE has 70% of market share doesn’t mean it’s the best in terms of quality Honda Civics out-sell BMWs and other luxury cars 4 or 5 to 1, but I think if you drive both cars you may find that a Civic is not a higher quality car than BMW
Misspelt or unknown words	addage, wewill, youhave, haveto	IE does not support the well known addage : time is money. IE’s security holes create an opening for spyware that slows down my computer, lost time and productivity

It is observed that the irrelevant aspects are either pronouns, common nouns, proper nouns, or unknown words. Pronouns do not represent aspects, and most of the common nouns are not related to debate targets. The syntactic rules cannot distinguish proper nouns that are actual aspects from the irrelevant ones. Also, there are cases of common nouns present in unrelated sentences. We perform

aspect pruning using word embeddings, to filter out such irrelevant aspects before further processing.

Word2vec embedding is used to build word vectors for aspects and targets. The distance between the word vectors is used to prune the aspects that are unrelated to the targets.

Word2vec Embedding: Word2vec model uses continuous bag-of-words and skip-gram methods to generate word vectors. Word vectors are learned from Google News dataset, which consists of almost 100 billion words. Word meaning and relationship between the words are encoded spatially. The spatial distance between the word vectors correspond to word similarity. We therefore generate word vectors for aspects and targets using word2vec and compute word vector similarity score. Since there are two targets in a debate, each aspect generates a two-dimensional feature vector from the similarity scores computed using the two targets. Once we have the feature vectors, we use them as features to build Gaussian Naive Bayes (GNB) and Support Vector Machine (SVM) aspect classifier. Upon comparison, GNB is able to predict related aspects better than SVM since the feature vectors are only two dimensional and GNB works well with continuous values. Hence, we use GNB for aspect pruning. The details of GNB implementation are presented in Sect. 4.

3.5 Aspect-Polarity Mapping

After getting the aspects related to debate targets, the opinion words in aspect-opinion pairs are substituted with the respective polarities from O_L to form aspect-polarity pairs (e.g., $\{Windows, bad\} \rightarrow \{Windows, -1\}$). There is a special case where negation words (e.g., not, no, and nor) are present before the opinion words. In such cases, the polarity of the opinion word present in O_L is reversed. Thus, we get the sentiment expressed on aspects of the post. Aspect-polarity pairs (AP) are used to find the opinions expressed on the targets.

3.6 Aspect-Target Mapping

In this step, we find the target preference of each aspect. To find the target preference, we use the word vector similarity score generated using Google word2vec. For each aspect a_i , we compute word2vec similarity between each relevant aspect a_i and the targets t_1 and t_2 . We also compute the difference ($diff$) between the similarity scores. The aspect-target preference dictionary (AT) for each a_i is determined based on the similarity and difference scores as follows:

$$AT[a_i] = \begin{cases} t_1, t_2 & \text{if } (diff < th) \\ t_1, & \text{if } (sim1 > sim2) \\ t_2, & \text{if } (sim2 > sim1) \end{cases}$$

where, $sim1$ and $sim2$ are the similarity score of each aspect with targets t_1 and t_2 respectively. If the difference in the similarity score is less than a threshold th , we consider that the aspect is related to both the targets. Otherwise, the aspect

prefers the target with the higher similarity score. The value of th is determined empirically using the four datasets described in Sect. 4. We tried different values of th (0.05, 0.10, 0.15, 0.20, 0.25, 0.30) and found that $th = 0.1$ gives the best result.

3.7 Detect Stance

The next step is to deduce the stance of the user. We articulate this problem as an ILP optimization problem. ILP is a powerful optimization technique, which optimizes a function based on some given constraints. We use the AT preference dictionary and the similarity information to calculate a weighted similarity score. This score is used to formulate the ILP equations. For calculating the weighted similarity score, we collect all the aspect-polarity pairs AP and the aspect-target preference dictionary AT . Let m be the number of aspect-polarity pairs present in a post. The weighted similarity score for each target-aspect pair (S_{ij}) is computed as follows:

$$S_{ij} = (AP(j) * sim(i, j) * NF(j) + diff(sim(i, j), sim(1 - i, j)) + T_i) * PT_j \quad (1)$$

where, $i = (0, 1)$, represents the debate targets, $j = \{a_1, a_2, a_3, \dots, a_m\}$ represents the relevant aspects present in the post, $AP(j) = \{-1, 0, 1\}$ is the polarity of aspect j in the post, $sim(i, j)$ is a continuous valued word2vec similarity score between the target i and the aspect j , and $diff(sim(i, j), sim(1 - i, j))$ is the absolute difference of the similarity scores of aspect j with each of the targets. The difference score is used to penalize those aspects that are common to both the targets and reward those aspects that are unique towards a target. T_i is a preference value that rewards when the aspect is also the target. This happens when a the post contains a target along with an associated opinion, such as “*Windows is obviously better than Mac*”. $NF(j)$ is the normalized frequency of the aspect j . It is used to give more weightage to those aspects that are present more frequently in the post corpus. This is because frequent aspects are generally more popular and well known. $NF(j)$ is computed as the ratio of the frequency of aspect j in the corpus to the frequency of all the aspects in the corpus. It is defined as follows:

$$NF(j) = \frac{|j|}{\sum_{p=1}^m |j_p|} \quad (2)$$

PT_j is the value of aspect preference over the target. If the aspect does not prefer the target, it gives a score of 0, thus ignoring the weighted score when the aspect prefers the opposite target. If the aspect prefers the target, then PT_j gets a value of 1. The preference information is obtained from AT dictionary. It is formulated as follows:

$$PT_j = \begin{cases} 1, & \text{if } (j \in AT[j]) \\ 0, & \text{otherwise} \end{cases}$$

ILP Formulation: The stance of the post is determined by the side that maximizes the ILP objective function. Given all the aspect-target instances of a

post, the objective function is formulated as follows:

$$\sum_{j=1}^m (S_{0j}p_j + S_{1j}q_j) \quad (3)$$

and it's constraints are,

$$p_j, q_j \in 0, 1, \forall j \quad (4)$$

$$p_j + q_j = 1, \forall j \quad (5)$$

$$p_j - p_{j-1} = 0, j \in \{2 \dots m\} \quad (6)$$

$$q_j - q_{j-1} = 0, j \in \{2 \dots m\} \quad (7)$$

where, p_j, q_j are boolean variables, Eq. 5 makes sure that the variables are mutually exclusive so that each aspect instance can prefer only one target. Equations 6 and 7 make sure that the user supports only one side and writes in support of that side for all aspects. This is based on the assumption that in a debate, the user is consistent about her stance of supporting single target and its aspects. So, for a particular post, all the values of p_j (or q_j) are made identical. This is a unique condition for debate posts unlike others such as forum posts, reviews, and tweets, where the user writes both good and bad points about a target.

4 Evaluation

In this section, we describe our dataset and evaluation metrics. We compare the performance of our proposed system, Debate Stance Classification using Word Embeddings (*DSCW*) with existing unsupervised debate classification approaches and show that our method can detect stance side more accurately.

4.1 Dataset

We consider the debate side annotated corpora used in [14]. This corpora consists of posts from four debate topics *Firefox vs. Internet Explorer (IE)*, *Opera vs. Firefox*, *SonyPs3 vs. Wii*, and *Windows vs. Mac*. Each of the dataset has manually annotated debate side preference. The datasets are summarized in Table 2.

Table 2. Details of debate post dataset.

Sl. No.	Debate topic	Number of posts
1	Firefox vs. Internet Explorer (IE)	169
2	Opera vs. Firefox	16
3	SonyPs3 vs. Wii	68
4	Windows vs. Mac	27

4.2 Stance Prediction Evaluation

Evaluation Metrics: We evaluate the stance prediction performance of our proposed method in terms of accuracy of debate stance classification. We consider all the post as relevant for debate classification. The accuracy of our debate classification method in classifying debate stance is measured in terms of precision(P), recall(R) and $F1$ score. The metrics are defined as follows:

$$P = \frac{\# \text{ correctly classified post}}{\# \text{ classified post}}; \quad R = \frac{\# \text{ correctly classified posts}}{\# \text{ relevant post}}; \quad F1 = \frac{2 * P * R}{P + R}$$

Baseline Methods: In order to evaluate the effectiveness of the proposed method, we conducted accuracy comparison with three baseline methods. We applied OpTopic, OpPMI, and OpProbability methods described in [14] on the four datasets and determine their stance prediction accuracy.

OpTopic finds target-polarity pairs from debate posts and counts the number of positive opinion words associated with each of the debate topic. A negative opinion word or a topic is counted as a positive opinion word for the opposing topic. Debate stance is assigned to the topic with more positive opinion word count.

OpPMI extracts terms (nouns) from debate posts and computes Pointwise Mutual Information (PMI) of the terms with each target. Then, the target-polarity pairs are found and the count of positive opinion words are computed. Debate stance is assigned to the side with more positive opinion word count as in OpTopic. Each term is then assigned to the target with higher PMI value. The polarity-target pair for each noun is then computed, to find terms which are closely related to the debate topics. In both methods, the post is assigned to the side with higher cumulative polarity-topic pair score.

In OpProbability, aspect-opinion pairs are extracted from debate posts and are converted to aspect-polarity pairs. Weblogs and forums related to the topics are mined to find aspect-target preference. An aspect-target preference probability table is then constructed from the mined corpus and a classifier is built using ILP to classify debate posts.

We compare the performance of the three baseline approaches with our two proposed methods: debate Opinion detection using Word Embedding (OpWE), and debate Opinion detection using Word Embedding and Aspect Pruning (OpWEAP). As mentioned in Sect. 3, OpWE uses word embeddings to build the aspect-target preference mapping (AT) by comparing the word embedding similarity scores between the aspects and each target. Then stance classification is done by applying ILP to weighted similarity score (S_{ij}) obtained from AT . Additionally, OpWEAP applies a Gaussian Naive Bayes aspect pruning classification to aspect-opinion pairs (as described in Sect. 3) before computing aspect-target preference dictionary (AT) to filter aspects that are not related to either of the targets.

Results: The result of the accuracy comparison of the five algorithms, namely OpTopic, OpPMI, OpProbability, OpWE and OpWEAP are presented in Table 3.

Table 3. Precision, Recall, and $F1$ measure of various debate classification methods.

Metric	Debate topic	OpTopic	OpPMI	OpProb	OpWE	OpWEAP
Precision	Firefox vs. IE	67.74	60.00	66.27	75.65	79.58
	Windows vs Mac	40.00	53.84	66.67	69.56	77.27
	SonyPs3 vs. Wii	80.00	46.15	61.11	68.96	70.90
	Opera vs. Firefox	33.33	100.00	100.00	75.00	81.82
Recall	Firefox vs.. IE	17.16	27.22	33.72	68.04	66.86
	Windows vs. Mac	7.04	25.93	37.03	59.25	62.96
	SonyPs3 vs. Wii	17.65	17.65	32.35	58.82	57.35
	Opera vs. Firefox	12.50	25.00	43.75	56.25	56.25
F1	Firefox vs. IE	27.38	37.45	44.70	71.64	75.41
	Windows vs. Mac	11.98	35.00	47.61	64.00	69.38
	SonyPs3 vs. Wii	28.92	25.53	42.30	63.48	63.41
	Opera vs. Firefox	18.18	40.00	60.86	64.29	66.67

From Table 3, we can see that among the three baselines, the OpProbability algorithm gives the highest precision and recall. The reason for this higher precision and recall compared to OpTopic and OpPMI is that it uses external knowledge to determine the aspect-target preference. OpProbability is able to learn aspect preference from the web corpus. Even this approach is limited by the availability of proper domain information from the web corpus. The web corpus may have limited number of pages or irrelevant pages.

As can be noted, OpTopic has very low recall in detecting debate stance. This is because it is a naive approach that relies only on opinions associated with the debate topic. The opinions of the aspects related to the topic are discarded. OpPMI is able to detect stance with higher recall as it associates each term with either of the target and opinion of each term is considered. However, the precision drops as not all the terms are aspects and a term be associated with both targets.

Compared to OpProbability our proposed two algorithm gives 10% higher precision and 30% higher recall. The reason for this very high recall is that we are able to detect stance in more number of posts because of better pre-processing, more external knowledge in the form of word-embedding, and a bigger opinion lexicon.

In the *Opera vs Firefox* dataset, two of the baseline algorithms give 100% precision, whereas our best method gives only 82%. The reason is the size of the dataset. This dataset has only 16 posts, and the algorithm is able to detect stance in only 4 out of the 16 posts. Since it detects correct stance in all the 4 posts, it has 100% precision. However, the recall of these algorithms is 25% and 43%, which is 13% lower compared to our methods.

OpWE works well in multiple domains since the word-embeddings are obtained from large text corpus and no domain specific corpus is required. Compared to OpWE, OpWEAP further improves the average precision by 5% and

F_1 score by 2.85%. The result suggests that pruning aspects irrelevant to the targets increases the classification accuracy. The reason for the increase is that it removes their influence in detecting stance in the ILP formulation.

4.3 Aspect Pruning Evaluation

In this section, we describe the GNB classifier used for pruning irrelevant aspects, and evaluate its effectiveness in determining topic related aspects. We also evaluate the use of the GNB classifier, trained on a domain, for transfer learning for pruning irrelevant aspects in other domains.

Gaussian Naive Bayes (GNB): Initially, we consider *Firefox vs. IE* debate posts to build the GNB aspect-target classifier. There are 552 aspects present in AOP of *Firefox vs. IE* debate posts. Three research scholars manually annotate the aspects as related or not related to either of the targets. The disagreements are resolved through mutual discussion. After manual annotation, it is found that 114 aspects are related to either of the targets and 438 aspects not related. Then, we perform a random split of the aspects into 370 (66.66%) training and 182 (33.33%) testing data. The 2-D similarity score feature vector is used to train the GNB classifier. After cross validation, the trained model gives an average accuracy of 81% in classifying the aspects. This trained model is used to classify whether a new aspect from the post is related to either of the debate target or not.

In general, manual labeling of aspects for each debate takes effort and is a time-consuming process. We, therefore, test if transfer learning can be applied on the GNB classifier for cross-domain classification of irrelevant aspects. For the purpose, the research scholars perform manual annotation of related aspects on the remaining three datasets. The above trained GNB classifier is applied to the remaining three datasets to remove irrelevant aspects. The results of the classifier are shown in Table 4, where the classifier is trained on only *Firefox vs. IE* debate dataset.

Table 4. Aspect pruning accuracy across all four debate datasets.

Measure	Firefox vs. IE	Windows vs. Mac	Sony Ps3 vs. Wii	Opera vs. Firefox
Classification accuracy (%)	81	84	87	84.20

We observe that the GNB classifier can detect target related aspects across multiple domains with high accuracy. It indicates that the classifier learns a function to discriminate whether an aspect is relevant or irrelevant based on the aspects similarity with the target topics. Since this function is independent of the targets or the aspects, it can be used for transfer learning across multiple debate topics, without lowering the classification accuracy.

5 Conclusion

This paper presents an unsupervised approach to perform debate stance classification using word embeddings. The use of word embeddings enables us to perform stance detection without using web corpus. Our results show that our approach can detect user stance significantly better than existing methods. In existing techniques, many of the aspects obtained are irrelevant to the debate topics. We introduce data cleaning, lexicon expansion, and aspect pruning for better opinion and aspect detection. We also train a supervised aspect classifier that can be used as transfer learning method to detect target related aspects across multiple domains. Our approach is scalable to two-sided debate posts across multiple domains as it does not require any domain specific information to perform stance detection.

References

1. Anand, P., Walker, M., Abbott, R., Tree, J.E.F., Bowmani, R., Minor, M.: Cats rule and dogs drool! classifying stance in online debate. In: Proceedings of the Workshop on Computational Approaches to Subjectivity and Sentiment Analysis, pp. 1–9. ACL (2011)
2. Burfoot, C., Bird, S., Baldwin, T.: Collective classification of congressional floor-debate transcripts. In: Proceedings of ACL HLT 2011, pp. 1506–1515. ACL (2011)
3. Ghosh, S., Anand, K., Rajanala, S., Reddy, A.B., Singh, M.: Unsupervised stance classification in online debates. In: Proceedings of the ACM India Joint International Conference on Data Science and Management of Data, pp. 30–36. ACM (2018)
4. Hu, M., Liu, B.: Mining opinion features in customer reviews. In: Proceedings of AAAI 2004, vol. 4, pp. 755–760 (2004)
5. Krejzl, P., Steinberger, J.: Uwb at semeval-2016 task 6: stance detection. In: Proceedings of SemEval@ NAACL-HLT, pp. 408–412 (2016)
6. Levy, O., Goldberg, Y.: Dependency-based word embeddings. In: Proceedings of ACL 2014, pp. 302–308 (2014)
7. Liu, B., Hu, M., Cheng, J.: Opinion observer: analyzing and comparing opinions on the web. In: Proceedings of WWW 2005, pp. 342–351. ACM (2005)
8. Liu, Q., Liu, B., Zhang, Y., Kim, D.S., Gao, Z.: Improving opinion aspect extraction using semantic similarity and aspect associations. In: Proceedings of AAAI 2016, pp. 2986–2992. AAAI Press (2016)
9. Lu, Y., Wang, H., Zhai, C., Roth, D.: Unsupervised discovery of opposing opinion networks from forum discussions. In: Proceedings of CIKM 2012, pp. 1642–1646. ACM (2012)
10. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of NIPS 2013, pp. 3111–3119 (2013)
11. Mohammad, S.M.: # emotional tweets. In: Proceedings of SemEval 2012, pp. 246–255. ACL (2012)
12. Mohammad, S.M., Turney, P.D.: NRC emotion lexicon. NRC Technical report (2013)

13. Pfohl, S.R., Triebe, O.: Stance detection for the fake news challenge with attention and conditional encoding. In: Stanford CS224d Deep Learning for NLP final project (2017)
14. Somasundaran, S., Wiebe, J.: Recognizing stances in online debates. In: Proceedings of ACL-IJCNLP 2009, pp. 226–234. ACL (2009)
15. Thomas, M., Pang, B., Lee, L.: Get out the vote: determining support or opposition from congressional floor-debate transcripts. In: Proceedings of EMNLP 2006, pp. 327–335. ACL (2006)
16. Walker, M.A., Anand, P., Abbott, R., Grant, R.: Stance classification using dialogic properties of persuasion. In: Proceedings of ACL 2012, pp. 592–596. ACL (2012)
17. Xu, L., Liu, K., Zhao, J., et al.: Joint opinion relation detection using one-class deep neural network. In: Proceedings of the COLING 2014, vol. 14, pp. 677–687 (2014)

Author Index

- Abel, Mara 177
Adachi, Jun 49
Aihara, Kenro 49
Ali, Chérif Arab 193
Almeida, Cassio 126
Almutairi, Nawal 163
Angryk, Rafal 371
- Bahrpeyma, Fouad 345
Barbosa, Simone 126
Bemarisika, Parfait 329
Ben HajKacem, Mohamed Aymen 317
Ben Hamadou, Hamdi 289
Ben N'cir, Chiheb-Eddine 317
Bodra, Jay 271
Borges, Heraldo 247
Bouadi, Tassadit 111
Braun, Peter 83
Brdar, Sanja 19
- Campisano, Riccardo 247
Carbonera, Joel Luís 177
Chakravarthy, Sharma 271
Chevalier, Max 289
Coenen, Frans 163
- Das, Soumyava 271
Djenouri, Youcef 204
Dures, Keith 163
- El Malki, Mohammed 289
Essoussi, Nadia 317
Ezeife, C. I. 70
- Filali Boubrahimi, Soukaina 371
Fiol-Gonzalez, Sonia 126
Fournier-Viger, Philippe 204, 231
Fujita, Hamido 231
- Ganibardi, Amine 193
Ghabry, Issam 302
Ghosh, Subrata 382
Ghrab, Amine 3
- Jelassi, M. Nidhal 302
Jha, Alok Kumar 34
Jouili, Salim 3
- Kawakatsu, Takaya 49
Kinoshita, Akira 49
Kiran, Rage Uday 231
Kitsuregawa, Masaru 355
Konjengbam, Anand 382
Kotni, Amulya 355
Krishna Reddy, P. 355
Kumar, Nagendra 382
- Leung, Carson K. 83
Li, Zhitian 231
Lin, Jerry Chun-Wei 204, 231
Liu, Qiankun 204
Lopes, Hélio 126
- Ma, Ruizhe 371
Martin, Arnaud 111
Masseglia, Florent 247
McCarren, Andrew 345
Mehmood, Qaiser 34
- Nakamura, Satoshi 216
Novović, Olivera 19
- Ogasawara, Eduardo 247
- Pacitti, Esther 247
Papadopoulos, Apostolos N. 19
Papanikolaou, Yannis 152
Pazdor, Adam G. M. 83
Péninou, André 289
Perosi, Fabio 247
Plant, Claudia 137
Porto, Fabio 247
- Rajan, K. S. 59
Rebholz-Schuhmann, Dietrich 34
Roantree, Mark 345
Romero, Oscar 3

Sahay, Ratnesh 34
Sankepally, Rohith Reddy 59
Santra, Abhishek 271
Schelling, Benjamin 137
Singh, Manish 382
Singh, Ritu 98
Skhiri, Sabri 3
Suzuki, Yu 216

Takasu, Atsuhiko 49
Tanaka, Hiroaki 216
Teste, Olivier 289
Toshniwal, Durga 98
Totohasina, André 329
Toyoda, Masashi 355

Truică, Ciprian-Octavian 19
Tsoumakas, Grigorios 152

Uday Kiran, R. 355

Waage, Tim 261
Wiese, Lena 261

Xiao, Ying 70

Yahia, Sadok Ben 302
Yoshino, Koichiro 216

Zhang, Ji 204
Zhang, Yiru 111