

Petra Perner (Ed.)

LNAI 10933

Advances in Data Mining

Applications and Theoretical Aspects

18th Industrial Conference, ICDM 2018
New York, NY, USA, July 11–12, 2018
Proceedings

 Springer

Lecture Notes in Artificial Intelligence

10933

Subseries of Lecture Notes in Computer Science

LNAI Series Editors

Randy Goebel

University of Alberta, Edmonton, Canada

Yuzuru Tanaka

Hokkaido University, Sapporo, Japan

Wolfgang Wahlster

DFKI and Saarland University, Saarbrücken, Germany

LNAI Founding Series Editor

Joerg Siekmann

DFKI and Saarland University, Saarbrücken, Germany

More information about this series at <http://www.springer.com/series/1244>

Petra Perner (Ed.)

Advances in Data Mining

Applications and Theoretical Aspects

18th Industrial Conference, ICDM 2018

New York, NY, USA, July 11–12, 2018

Proceedings

Editor
Petra Perner
Institute of Computer Vision and Applied
Computer Sciences
Leipzig
Germany

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Artificial Intelligence
ISBN 978-3-319-95785-2 ISBN 978-3-319-95786-9 (eBook)
<https://doi.org/10.1007/978-3-319-95786-9>

Library of Congress Control Number: 2018947574

LNCS Sublibrary: SL7 – Artificial Intelligence

© Springer International Publishing AG, part of Springer Nature 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.

Printed on acid-free paper

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

Preface

The 18th event of the Industrial Conference on Data Mining ICDM was held in New York again (www.data-mining-forum.de) under the umbrella of the World Congress on Frontiers in Intelligent Data and Signal Analysis, DSA 2018 (www.worldcongressdsa.com).

After the peer-review process, we accepted 25 high-quality papers for oral presentation. The topics range from theoretical aspects of data mining to applications of data mining, such as in multimedia data, in marketing, in medicine and agriculture, and in process control, industry, and society. Extended versions of selected papers will appear in the international journal *Transactions on Machine Learning and Data Mining* (www.ibai-publishing.org/journal/mldm).

In all, 20 papers were selected for poster presentations and six for industry paper presentations, which are published in the ICDM Poster and Industry Proceedings by ibai-publishing (www.ibai-publishing.org).

The tutorial days rounded up the high quality of the conference. Researchers and practitioners got an excellent insight in the research and technology of the respective fields, the new trends, and the open research problems that we would like to study further.

A tutorial on Data Mining, a tutorial on Case-Based Reasoning, a tutorial on Intelligent Image Interpretation and Computer Vision in Medicine, Biotechnology, Chemistry and Food Industry, and a tutorial on Standardization in Immunofluorescence were held before and in between the conferences of DSA 2018.

We would like to thank all reviewers for their highly professional work and their effort in reviewing the papers.

We also thank the members of the Institute of Applied Computer Sciences, Leipzig, Germany (www.ibai-institut.de), who handled the conference as secretariat. We appreciate the help and understanding of the editorial staff at Springer, and in particular Alfred Hofmann, who supported the publication of these proceedings in the LNAI series.

Last, but not least, we wish to thank all the speakers and participants who contributed to the success of the conference. We hope to see you in 2019 in New York at the next World Congress on Frontiers in Intelligent Data and Signal Analysis, DSA 2019 (www.worldcongressdsa.com), which combines under its roof the following three events: International Conferences Machine Learning and Data Mining, MLDM (www.mldm.de), the Industrial Conference on Data Mining, ICDM (www.data-mining-forum.de), and the International Conference on Mass Data Analysis of Signals and Images in Medicine, Biotechnology, Chemistry, Biometry, Security, Agriculture, Drug Discovery and Food Industry, MDA (www.mda-signals.de), as well as the workshops, and tutorials.

Organization

Chair

Petra Perner

IBal Leipzig, Germany

Program Committee

Ajith Abraham

Machine Intelligence Research Labs (MIR Labs), USA

Brigitte Bartsch-Spörl

BSR Consulting GmbH, Germany

Orlando Belo

University of Minho, Portugal

Bernard Chen

University of Central Arkansas, USA

Antonio Dourado

University of Coimbra, Portugal

Jeroen de Bruin

Medical University of Vienna, Austria

Stefano Ferilli

University of Bari, Italy

Geert Gins

KU Leuven, Belgium

Warwick Graco

ATO, Australia

Aleksandra Gruca

Silesian University of Technology, Poland

Hartmut Ilgner

Council for Scientific and Industrial Research,
South Africa

Pedro Isaias

Universidade Aberta (Portuguese Open University),
Portugal

Piotr Jedrzejowicz

Gdynia Maritime University, Poland

Martti Juhola

University of Tampere, Finland

Janusz Kacprzyk

Polish Academy of Sciences, Poland

Mehmed Kantardzic

University of Louisville, USA

Eduardo F. Morales

INAOE, Ciencias Computacionales, Mexico

Samuel Noriega

Universitat de Barcelona Spain

Juliane Perner

Cancer Research, Cambridge Institutes, UK

Armand Prieditris

Newstar Labs, USA

Rainer Schmidt

University of Rostock, Germany

Victor Sheng

University of Central Arkansas, USA

Kaoru Shimada

Section of Medical Statistics, Fukuoka Dental College,
Japan

Gero Szepannek

Stralsund University, Germany

Markus Vattulainen

Tampere University, Finland

Additional Reviewers

Dimitrios Karras
Calin Ciufudean
Valentin Brimkov
Michelangelo Ceci
Reneta Barneva
Christoph F. Eick
Thang Pham
Giorgio Giacinto
Kamil Dimililer

Contents

An Adaptive Oversampling Technique for Imbalanced Datasets	1
<i>Shaukat Ali Shahee and Usha Ananthakumar</i>	
From Measurements to Knowledge - Online Quality Monitoring and Smart Manufacturing	17
<i>Satu Tamminen, Henna Tiensuu, Eija Ferreira, Heli Helakoski, Vesa Kyllönen, Juha Jokisaari, and Esa Puukko</i>	
Mining Sequential Correlation with a New Measure	29
<i>Mohammad Fahim Arefin, Maliha Tashfia Islam, and Chowdhury Farhan Ahmed</i>	
A New Approach for Mining Representative Patterns	44
<i>Abeda Sultana, Hosneara Ahmed, and Chowdhury Farhan Ahmed</i>	
An Effective Ensemble Method for Multi-class Classification and Regression for Imbalanced Data	59
<i>Tahira Alam, Chowdhury Farhan Ahmed, Sabit Anwar Zahin, Muhammad Asif Hossain Khan, and Maliha Tashfia Islam</i>	
Automating the Extraction of Essential Genes from Literature.	75
<i>Ruben Rodrigues, Hugo Costa, and Miguel Rocha</i>	
Rise, Fall, and Implications of the New York City Medallion Market	88
<i>Sherraina Song</i>	
An Intelligent and Hybrid Weighted Fuzzy Time Series Model Based on Empirical Mode Decomposition for Financial Markets Forecasting	104
<i>Ruixin Yang, Junyi He, Mingyang Xu, Haoqi Ni, Paul Jones, and Nagiza Samatova</i>	
Evolutionary DBN for the Customers' Sentiment Classification with Incremental Rules	119
<i>Ping Yang, Dan Wang, Xiao-Lin Du, and Meng Wang</i>	
Clustering Professional Baseball Players with SOM and Deciding Team Reinforcement Strategy with AHP.	135
<i>Kazuhiro Kohara and Shota Enomoto</i>	
Data Mining with Digital Fingerprinting - Challenges, Chances, and Novel Application Domains	148
<i>Matthias Vodel and Marc Ritter</i>	

Categorization of Patient Diseases for Chinese Electronic Health Record Analysis: A Case Study	162
<i>Junmei Zhong, Xiu Yi, De Xuan, and Ying Xie</i>	
Dynamic Classifier and Sensor Using Small Memory Buffers	173
<i>R. Gelbard and A. Khalemsky</i>	
Speeding Up Continuous kNN Join by Binary Sketches.	183
<i>Filip Nalepa, Michal Batko, and Pavel Zezula</i>	
Mining Cross-Level Closed Sequential Patterns.	199
<i>Rutba Aman and Chowdhury Farhan Ahmed</i>	
An Efficient Approach for Mining Weighted Sequential Patterns in Dynamic Databases	215
<i>Sabrina Zaman Ishita, Faria Noor, and Chowdhury Farhan Ahmed</i>	
A Decision Rule Based Approach to Generational Feature Selection	230
<i>Wiesław Paja</i>	
A Partial Demand Fulfilling Capacity Constrained Clustering Algorithm to Static Bike Rebalancing Problem.	240
<i>Yi Tang and Bi-Ru Dai</i>	
Detection of IP Gangs: Strategically Organized Bots	254
<i>Tianyue Zhao and Xiaofeng Qiu</i>	
Medical AI System to Assist Rehabilitation Therapy	266
<i>Takashi Isobe and Yoshihiro Okada</i>	
A Novel Parallel Algorithm for Frequent Itemsets Mining in Large Transactional Databases	272
<i>Huan Phan and Bac Le</i>	
A Geo-Tagging Framework for Address Extraction from Web Pages.	288
<i>Julia Efremova, Ian Endres, Isaac Vidas, and Ofer Melnik</i>	
Data Mining for Municipal Financial Distress Prediction	296
<i>David Alaminos, Sergio M. Fernández, Francisca García, and Manuel A. Fernández</i>	
Prefix and Suffix Sequential Pattern Mining	309
<i>Rina Singh, Jeffrey A. Graves, Douglas A. Talbert, and William Eberle</i>	
Author Index	325



An Adaptive Oversampling Technique for Imbalanced Datasets

Shaukat Ali Shahee and Usha Ananthakumar^(✉)

Indian Institute of Technology Bombay, Mumbai 400076, India
shaukatali.shahee@iitb.ac.in, usha@som.iitb.ac.in

Abstract. Class imbalance is one of the challenging problems in classification domain of data mining. This is particularly so because of the inability of the classifiers in classifying minority examples correctly when data is imbalanced. Further, the performance of the classifiers gets deteriorated due to the presence of imbalance within class in addition to between class imbalance. Though class imbalance has been well addressed in literature, not enough attention has been given to within class imbalance. In this paper, we propose a method that can adaptively handle both between-class and within-class imbalance simultaneously and also that can take into account the spread of the data in the feature space. We validate our approach using 12 publicly available datasets and compare the classification performance with other existing oversampling techniques. The experimental results demonstrate that the proposed method is statistically superior to other methods in terms of various accuracy measures.

Keywords: Classification · Imbalanced dataset · Oversampling
Model based clustering · Lowner John ellipsoid

1 Introduction

In data mining literature, class imbalance problem is considered to be quite challenging. The problem arises when the class of interest contains a relatively lower number of examples compared to other class examples. In this study, the minority class, the class of interest is considered positive and the majority class is considered negative. Recently, several authors have addressed this problem in various real life domains including customer churn prediction [6], financial distress prediction [10], employee churn prediction [39], gene regulatory network reconstruction [7] and information retrieval and filtering [35]. Previous studies have shown that applying classifiers directly to imbalance dataset results in poor performance [34, 41, 43]. One of the possible reasons for the poor performance is skewed class distribution because of which the classification error gets dominated by the majority class. Another kind of imbalance is referred to as *within-class* imbalance which pertains to the state where a class composes of different number of sub-clusters (sub-concepts) and these sub-clusters in turn, containing different number of examples.

In addition to class imbalance, *small disjuncts*, *lack of density*, *overlapping between classes* and *noisy examples* also deteriorate the performance of the classifiers [2, 28–30, 36]. The *between-class* imbalance along with *within-class* imbalance is an instance of *problem of small disjuncts* [26]. Literature presents different ways of handling class imbalance such as data preprocessing, algorithmic based, cost-based methods and ensemble of classifier sampling methods [12, 17]. Though no method is superior in handling all imbalanced problems, sampling based methods have shown great capability as they attempt to improve data distribution rather than the classifier [3, 8, 23, 42]. Sampling method is a preprocessing technique that modifies the imbalanced data to a balanced data using some mechanism. This is generally carried out by either increasing the minority class examples called as oversampling or by decreasing the majority examples, referred to as undersampling [4, 13]. It is not advisable to undersample the majority class examples if minority class has complete rarity [40]. The current literature available on simultaneous *between-class* imbalance and *within-class* imbalance is limited.

In this paper, an adaptive method for handling between class imbalance and within class imbalance simultaneously based on an oversampling technique is proposed. It also factors in the scatter of data for improving the accuracy of both the classes on the test set. Removing between class imbalance and within class imbalance simultaneously helps the classifier to give equal importance to all the sub-clusters, and adaptively increasing the size of sub-clusters handles the randomness in the dataset. Generally, classifier minimizes the total error, and removal of between class imbalance and within class imbalance helps the classifier in giving equal weight to all the sub-clusters irrespective of the classes thus resulting in increased accuracy of both the classes. Neural network is one such classifier and is being used in this study. The proposed method is validated on publicly available data sets and compared with well known existing oversampling techniques. Section 2 discusses the proposed method and analysis on publicly available data sets is presented in Sect. 3. Finally, Sect. 4 concludes the paper with future work.

2 An Adaptive Oversampling Technique

The approach in this proposed method is to oversample the examples in such a way that it helps the classifier in increasing the classification accuracy on the test set.

The proposed method is based on two challenging aspects faced by the classifiers in case of imbalanced data sets. First one is the case of the loss function, where the majority class dominates the minority class and thus eventually, minimization of the loss function is largely due to minimization of the majority class. Because of this, the decision boundary between the classes does not get shifted towards the minority class. Removing the between class and within class imbalance helps in removing the dominance of the majority class.

Another challenge faced by the classifiers is the accuracy of the classifiers on the test set. Due to the randomness of data, if the test example lies in the

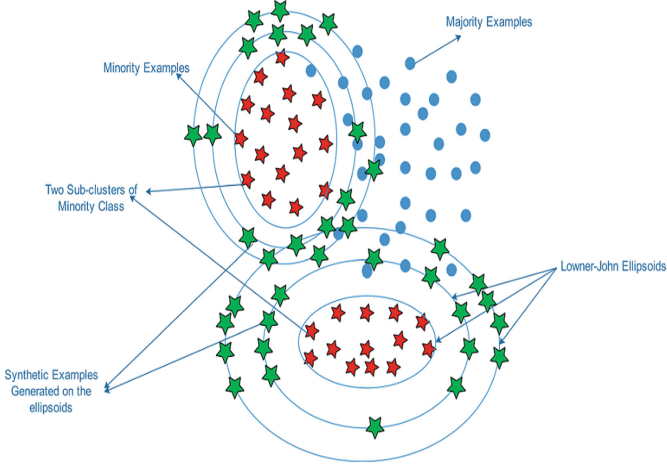


Fig. 1. Synthetic minority class examples generation on the peripheral of Lowner John ellipsoids

outskirts of the sub-clusters, there is a need to adjust the decision boundary to minimize misclassification. This is achieved by expanding the size of the sub-cluster in order to cope with such test examples. Now the question is, what is the surface of the sub-clusters and how far the sub-clusters should be expanded. To answer this, we use minimum volume ellipsoid that contains the dataset known as *Lowner John ellipsoid* [33]. We adaptively increase the size of the ellipsoid and synthetic examples are generated on the surface of the ellipsoid. One such instance is shown in Fig. 1 where minority class examples are denoted by stars and majority class examples by circle.

In the proposed method, the first step is data cleaning where the noisy examples are removed from the dataset as this helps in reducing the oversampling of noisy examples. After data cleaning, the concept is detected by using model based clustering and the boundary of each of the clusters is determined by *Lowner John ellipsoid*. Subsequently, the number of examples to be oversampled is determined based on the complexity of sub-clusters and synthetic data are generated on the peripheral of the ellipsoid. Following section elaborates the proposed method in detail.

2.1 Data Cleaning

In data cleaning process, the proposed method removes the noisy examples in the dataset. An example is considered as noisy if it is surrounded by all the examples of other class as defined in [3]. The number of examples is taken to be 5 in this study as also being considered in other studies including [3, 32].

2.2 Locating Sub-clusters

Model based clustering [16] is used with respect to minority class to identify the sub-clusters (or sub-concepts) present in the dataset. We have used MCLUST [15] for implementing the model based clustering. MCLUST is a *R* package that implements the combination of hierarchical agglomerative clustering, Expectation Maximization (EM) and Bayesian Information criterion (BIC) for comprehensive cluster analysis.

2.3 Structure of Sub-clusters

The structure of sub-clusters can be obtained using eigenvalues and eigenvector. Eigenvectors gives the shape of sub-cluster and size is given by eigenvalues. Let $X = \{x_1, x_2, \dots, x_m\}$ be a dataset having m examples and n features. Let the mean vector of X be μ and the covariance matrix computed by $\Sigma = E[(X - \mu)(X - \mu)^T]$. The eigenvalues (λ) and eigenvectors v of the covariance matrix Σ are found such that $\Sigma v = \lambda v$.

2.4 Identifying the Boundary of Sub-clusters

For each of the sub-clusters, *Lawner-John ellipsoid* is obtained as given by [33]. This is a minimum volume ellipsoid that contains the convex hull of $C = \{x_1, x_2, \dots, x_m\} \subseteq R^n$. The general equation of ellipsoid is

$$\varepsilon = \{v \mid \|Av + b\|_2 \leq 1\} \quad (1)$$

We assume that $A \in S_{++}^n$ is a positive definite matrix where the volume of ε is proportional to $\det A^{-1}$. The problem of computing the minimum volume ellipsoid containing C can be expressed as

$$\begin{aligned} & \text{minimize} && \log \det A^{-1} \\ & \text{subject to} && \|Ax_i + b\|_2 \leq 1, \quad i = 1, \dots, m. \end{aligned} \quad (2)$$

We use *CVX* [21], a Matlab-based modeling system for solving this optimization problem.

2.5 Synthetic Data Generation

The synthetic data generation is based on the following three steps

1. In the first step, the proposed method determines the number of examples to be oversampled per cluster. The number of minority class examples to be oversampled is computed using Eq. (3).

$$N = TC0 - TC1 \quad (3)$$

where N is the number of minority class examples to be oversampled, $TC0$ is the total number of examples of majority class *class 0* and $TC1$ is the total number of examples of *class 1*.

It then computes the complexity of sub-clusters based on the number of danger zone examples. An example is called a danger zone example or a borderline example if an example under consideration is surrounded by more than 50% examples of other class as also being considered in other studies including [23]. That is, if k is the number of nearest neighbors under consideration, an example being a danger zone example implies $k/2 \leq z < k$ where z is the number of other class examples among the k nearest neighbor examples. For example, Fig. 2 shows two sub-clusters of minority class having 4 and 2 danger zone examples. In this study, we consider $k = 5$ as in [3]. Let $c_1, c_2, c_3, \dots, c_q$ be the number of danger zone examples present in the sub-clusters $1, 2, \dots, q$ respectively. The number of examples to be oversampled in the sub-cluster i is given by

$$n_i = \frac{c_i * N}{\sum_{i=1}^q c_i} \quad (4)$$

- Having determined the number of examples to be oversampled, the next task is to weigh the danger zone examples in accordance with the direction of the ellipsoid and its distance from the centroid. These weights are computed with respect to the eigenvectors of the variance-covariance matrix of the dataset. For example, consider Fig. 3 where A and B denote the danger zone examples. Here we compute the inner product between danger zone examples A and B with the eigenvectors $EVec1$ and $EVec2$ that form acute angles with the danger zone examples. The weight of A , $W(A)$ is computed as

$$W(A) = \langle A, EVec1 \rangle + \langle A, EVec2 \rangle \quad (5)$$

Similarly the weight of B , $W(B)$ is computed as

$$W(B) = \langle B, EVec1 \rangle + \langle B, EVec2 \rangle \quad (6)$$

Thus, when data is n dimensional, the total weight of the b_k^{th} danger zone example w_k is

$$w_k = \sum_{i=1}^n \langle b_k, e_i \rangle \quad (7)$$

where e_i is the eigenvector.

- In each of the sub-clusters, synthetic examples are generated on the *Lowner John ellipsoid* by linear extrapolation of the selected danger zone example where the selection of danger zone example is carried out with respect to the weights obtained in step 2. Here

$$P(b_k) = \frac{w_k}{\sum_{i=1}^{c_i} w_i} \quad (8)$$

where $P(b_k)$ is the probability of selecting danger zone example b_k and w_k is the weight of k^{th} danger zone example present in the sub-cluster c_i . The selected danger zone example is extrapolated and a synthetic example is generated on the *Lowner John ellipsoid* at the point of intersection of the

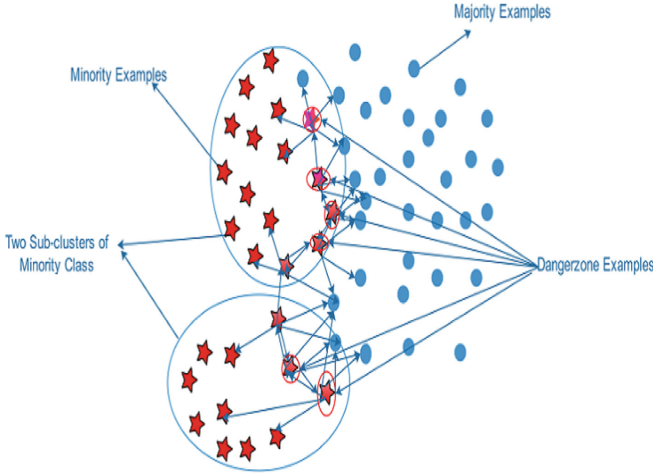


Fig. 2. Illustration of danger zone examples of minority class sub-clusters

extrapolated vector with *Lowner John ellipsoid*. Let the centroid of the ellipsoid be $center = -A^{-1} * b$ and if b_k is the danger zone example selected based on the probability distribution given by Eq. (8), the vector $v = b_k - center$ is extrapolated by ‘r’ units to intersect with the ellipsoid and the synthetic example s_t thus generated is given by

$$s_t = center + \frac{(r + C) * v}{\|v\|} \tag{9}$$

where C controls the expansion of the ellipsoid.

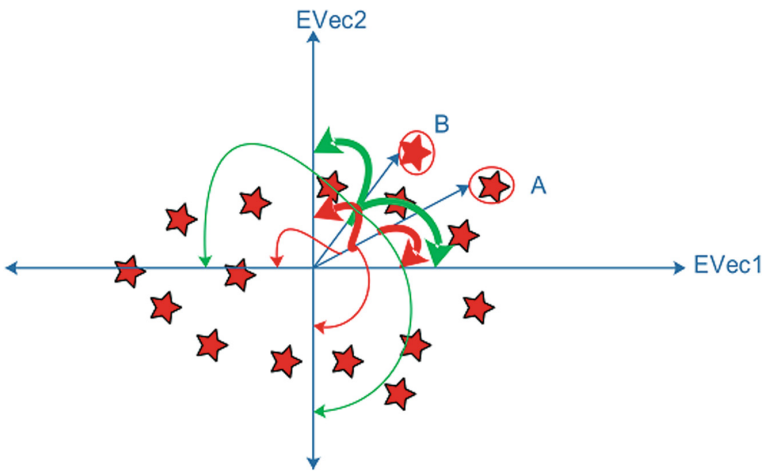


Fig. 3. Illustration of danger zone examples A & B of minority class forming acute angle with eigenvector in bold line

The whole procedure of the algorithm is explained in Algorithm 1.

Algorithm 1. An Adaptive Oversampling Technique for Imbalanced Data sets

Input: Training dataset: $S = \{X_i, y_i\}, i = 1, \dots, m; X_i \in R^n$ and $y_i \in \{0, 1\}$ Positive class: $S^+ = \{X_i^+, y_i^+\}, i = 1, \dots, m^+$; Negative class: $S^- = \{X_i^-, y_i^-\}, i = 1, \dots, m^-$; $S = S^+ \cup S^-; m = m^+ + m^-$ and No. of examples to be oversampled: $N = m^- - m^+$

Output: Oversampled Dataset

- 1: Clean the training set
 - 2: Apply Model-Based clustering on S^+ , return $\{smin_1, \dots, smin_q\}$ sub-clusters.
 - 3: **for each minority sub-cluster** $smin_i, 1 \leq i \leq q$ **do**
 - 4: $B_i \leftarrow$ DANGERZONEEXAMPLE($smin_i$) //Return list of danger zone examples
 - 5: **end for**
 - 6: **for each minority sub-cluster** $smin_i, 1 \leq i \leq q$ **do**
 - 7: $n_i = \frac{length(B_i) * N}{\sum_{n=1}^q length(B_i)}$ **for** $i = 1, \dots, q$ // No of examples to oversample in sub-cluster i
 - 8: **end for**
 - 9: **for** $i = 1$ **to** q **do**
 - 10: $\mu_i \leftarrow$ MEAN($smin_i$)
 - 11: $\Sigma_i \leftarrow$ COV($smin_i$)
 - 12: Compute the Lowner John ellipsoids of $smin_i$ as given in Subsect.2.4 gives A and b
 - 13: The eigenvectors v_1, \dots, v_n and eigenvalues $\lambda_1, \dots, \lambda_n$ of the covariance matrix Σ_i of dataset in sub-clusters $smin_i$ is computed by $\Sigma v_i = \lambda_i v$
 - 14: **for** $j = 1$ **to** $length(B_i)$ **do**
 - 15: $b_j \leftarrow B_i[j]$
 - 16: $w_j = 0$
 - 17: **for** $t = 1$ **to** n **do**
 - 18: $weight = \langle b_j, v_t \rangle$
 - 19: **if** $weight \geq 0$ **then**
 - 20: $w_j = w_j + weight$
 - 21: **end if**
 - 22: **end for**
 - 23: **end for**
 - 24: $p(b_j) = \frac{w_j}{\sum_{n=1}^{length(B_i)} w_n}$ // Compute the prob. dist of danger zone examples
 - 25: $NewSyntheticExample = \Phi$
 - 26: **for** $t = 1$ **to** n_i **do**
 - 27: Select the danger zone example based b_i based on step 24
 - 28: Synthetic example s_t has been generated as given in equation (9)
 - 29: $NewSyntheticExample = NewSyntheticExample \cup \{s_t\}$
 - 30: **end for**
 - 31: $oversample_i = smin_i \cup NewSyntheticExample$
 - 32: **end for**
 - 33: $Oversampled\ Dataset = \bigcup_{i=1}^q oversample_i$
-

3 Experiments

3.1 Data Sets

We evaluate the proposed method on 12 publicly available datasets which have skewed class distribution available on the KEEL dataset [1] repository. As *yeast* and *pageblocks* data sets have multiple classes, we have suitably transformed the data sets to two classes to meet our needs of binary class problem. In case of *yeast* dataset, it has 1484 examples and 10 classes {MIT, NUC, CYT, ME1, ME2, ME3, EXC, VAC, POX, ERL}. We choose ME3 as the minority class and the remaining are combined to form the majority class. In case of *pageblocks* dataset, it has 548 examples and 5 classes {1, 2, 3, 4, 5}. We choose 1 as majority class and the rest as the minority class. Minority class is chosen in both the data sets in such a way that it contains reasonable number of examples to identify the presence of sub-concepts and also to maintain the imbalance with respect to the majority class. The rest of the data sets were taken as they are. Table 1 represents the characteristics of various data sets used in the analysis.

Table 1. The data sets

Data sets	Total exp	Minority exp	Majority exp	No. attribute
glass1	214	76	138	9
pima	768	268	500	8
glass0	214	70	144	9
yeast1	1484	429	1055	8
vehicle2	846	218	628	18
ecoli1	336	77	259	7
yeast	1484	163	1321	8
glass6	214	29	185	9
yeast3	1484	163	1321	8
yeast-0-5-6-7-9_vs_4	528	51	477	8
yeast-0-2-5-7-9_vs_3-6-8	1004	99	905	8
pageblocks	548	56	492	10

3.2 Assessment Metrics

Traditionally, performance of classifiers is evaluated based on the *accuracy* and *error rate* as defined in (10). However, in case of the imbalanced dataset, the accuracy measure is not appropriate as it does not differentiate misclassification between the classes. Many studies address this shortcoming of accuracy measure with regard to imbalanced dataset [9, 14, 20, 31, 37]. To deal with class imbalance, various metric measures have been proposed in the literature that is based on the confusion matrix shown in Table 2.

Table 2. Confusion matrix

Classifier output		True class	
		p	n
P	TP	FP	
	FN	TN	

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

$$Error\ rate = 1 - Accuracy$$

These *confusion matrix* based measures described by [25] for imbalanced learning problem are *precision*, *recall*, *F-measure* and *G-mean*. These measures are defined as

$$Precision = \frac{TP}{TP + FP} \quad (11)$$

$$Recall = \frac{TP}{TP + FN} \quad (12)$$

$$F\text{-Measure} = \frac{(1 + \beta^2)Recall * Precision}{\beta^2 * Recall + Precision} \quad (13)$$

Here β is a non-negative parameter that controls the influence of precision and recall. In this study, we set $\beta = 1$ implying that precision and recall are equally important.

$$G\text{-Mean} = \sqrt{\frac{TP}{TP + FN} \frac{TN}{TN + FP}} \quad (14)$$

Another popular technique for evaluation of classifiers under imbalance domain is the Receiving Operating Characteristic (ROC) curve [37]. ROC curve is a graphical representation of the performance of the classifier by plotting *TP rates* versus *FP rates* over possible threshold values. The TP rates and FP rates are defined as

$$TP\ rate = \frac{TP}{TP + FN} \quad (15)$$

$$FP\ rate = \frac{FP}{FP + TN} \quad (16)$$

The quantitative representation of a ROC curve is the area under this curve and is called AUC [5, 27]. For the purpose of evaluation, we use *AUC* measure as it is independent of the distribution of positive class and negative class examples and hence this metric is not overwhelmed by the majority class examples. Apart from this, we have also considered *F-Measure* for both minority and majority class and *G-Mean* for comparative purposes.

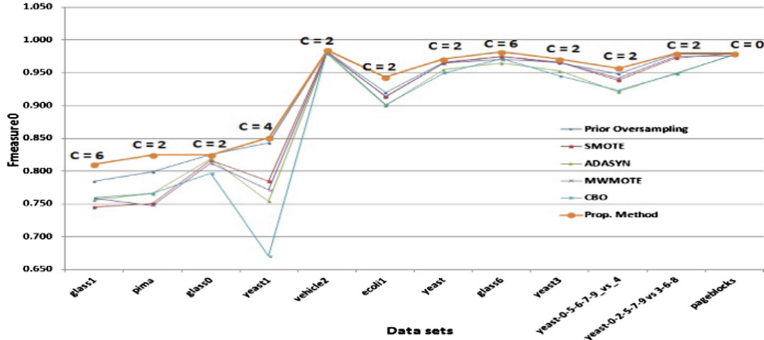


Fig. 4. Results of F -measure of majority class for various methods with the best one being highlighted.

3.3 Experimental Settings

In this work, we have used the feed-forward neural network with backpropagation. The structure of the network is such that it has input layers with the number of neurons being equal to the number of features. The number of neurons in the output layer is one as it is a binary classification problem. The number of neurons in the hidden layer is the average of the number of features and number of classes [22]. The activation function used at each neuron is the sigmoid function with learning rate 0.3.

We compare our proposed method with well known existing oversampling methods such as *SMOTE* [8], *ADASYN* [24], *MWMOTE* [3] and *CBO* [30]. We use default parameter settings for these oversampling techniques. In case of *SMOTE* [8], the number of nearest neighbor k is set to 5. In case of *ADASYN* [24], the number of nearest neighbor k is 5 and desired level of balance is 1. In case of *MWMOTE* [3], the number of neighbors used for predicting noisy minority class examples is $k1 = 5$, the number of nearest neighbors used to find majority class examples is $k2 = 3$, the percentage of original minority class examples used in generating synthetic examples is $k3 = |S_{min}|/2$, the number of clusters in the method is $Cp = 3$ and smoothing and rescaling values of different scaling factors are $Cf(th) = 5$ and $C_{MAX} = 2$ respectively.

3.4 Results

The results of 12 data sets for metric measures F -measure of majority and minority class, G-mean and AUC are shown in Figs. 4, 5, 6 and 7. It is enough to show F -measure rather than explicitly showing *Precision* and *Recall* because F -measure integrates *Precision* and *Recall*. We used 5-fold stratified cross-validation technique that runs 5 independent times and average of this is presented in Figs. 4, 5, 6 and 7. In 5-fold stratified cross-validation technique, a dataset is divided into 5 folds having an equal proportion of the classes. Among the 5 folds, one fold is considered as the test set and the remaining 4 folds are

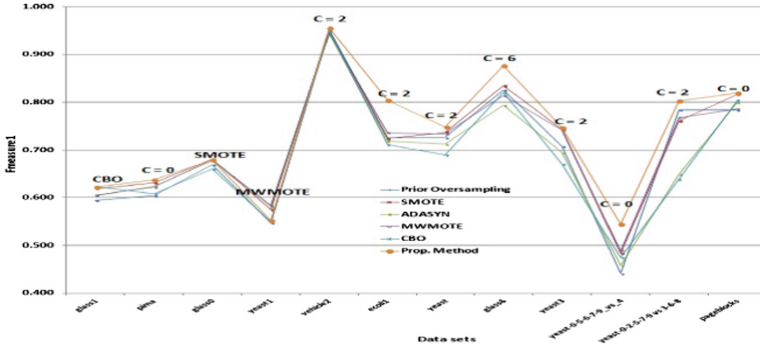


Fig. 5. Results of F -measure of minority class for various methods with the best one being highlighted.

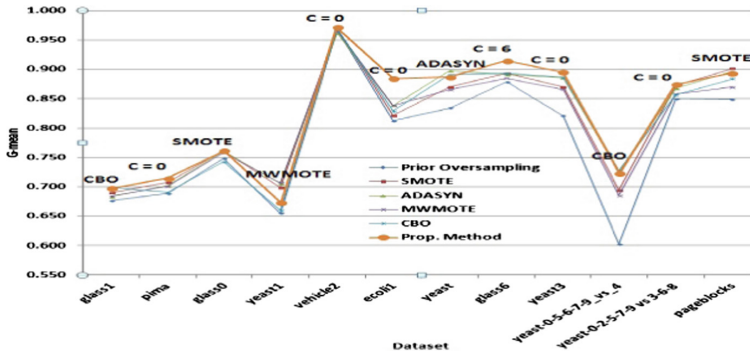


Fig. 6. Results of G -mean for various methods with the best one being highlighted.

combined and considered as the training set. Oversampling is carried out only on the training set and not on the test set in order to obtain unbiased estimates of the model for future prediction.

Figure 4 shows the results of F -measure of majority class. It is clear from the figure that the proposed method outperforms the other oversampling methods for different values of C . In this study, we consider $C \in \{0, 2, 4, 6\}$ where C controls the expansion of the ellipsoid. $C = 0$ gives the minimum volume *Lowner-John ellipsoid* and $C = 2$ means the size of ellipsoid increases by 2 units. The results of F measure1 is shown in Fig. 5. From the figure it is clear that the proposed method outperforms the other methods except in case of data sets *glass1*, *glass0* and *yeast1* where *CBO*, *SMOTE* and *MWMOTE* perform slightly better. Similarly, the results in case of G -mean and AUC are shown in Figs. 6 and 7 respectively. The method yielding the best result is highlighted in all the figures.

To compare the proposed method with other oversampling methods, we carried out non-parametric tests as suggested in the literature [11, 18, 19]. Wilcoxon

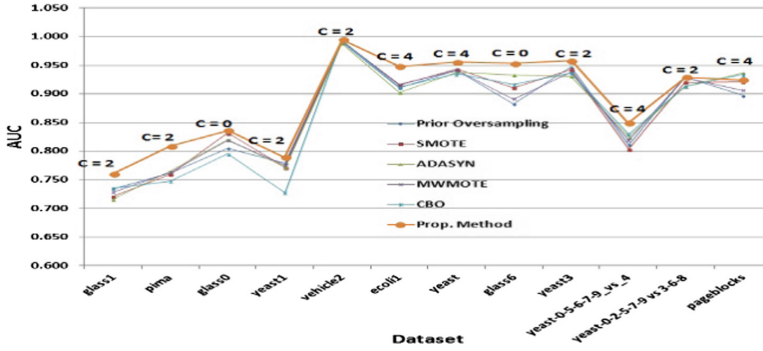


Fig. 7. Results of AUC for various methods with the best one being highlighted.

Table 3. Summary of Wilcoxon signed rank test between our proposed method and other methods

Methods	Proposed method	Metric measure	Hypothesis
Prior oversampling	p_value = 0.003204	F-measure of majority	H_0 rejected
	p_value = 0.002516	F-measure of minority	H_0 rejected
	p_value = 0.0004883	G-mean	H_0 rejected
	p_value = 0.003857	AUC	H_0 rejected
SMOTE	p_value = 0.002516	F-measure of majority	H_0 rejected
	p_value = 0.02061	F-measure of minority	H_0 rejected
	p_value = 0.07733	G-mean	Fail to reject H_0
	p_value = 0.0004883	AUC	H_0 rejected
ADASYN	p_value = 0.0004883	F-measure of majority	H_0 rejected
	p_value = 0.009766	F-measure of minority	H_0 rejected
	p_value = 0.2298	G-mean	Fail to reject H_0
	p_value = 0.004164	AUC	H_0 rejected
MWMOTE	p_value = 0.002478	F-measure of majority	H_0 rejected
	p_value = 0.01344	F-measure of minority	H_0 rejected
	p_value = 0.02531	G-mean	H_0 rejected
	p_value = 0.003857	AUC	H_0 rejected
CBO	p_value = 0.0004883	F-measure of majority	H_0 rejected
	p_value = 0.0009766	F-measure of minority	H_0 rejected
	p_value = 0.01669	G-mean	H_0 rejected
	p_value = 0.001465	AUC	H_0 rejected

signed-rank non-parametric test [38] is carried out on *F-measure of majority class*, *F-measure of minority class*, *G-Mean* and *AUC*. The null and alternative hypothesis are as follows:

- H_0 : The median difference is zero
- H_1 : The median difference is positive.

This test computes the difference in the respective measure between the proposed method and the method compared with it and ranks the absolute differences. Let $W+$ be the sum of the ranks with positive differences and $W-$ be the sum of the ranks with negative differences. The test statistic is defined as $W = \min(W+, W-)$. Since there are 12 data sets, the *W value* should be less than 17 (critical value) at a significance level of 0.05 to reject H_0 [38]. Table 3 shows the *p-values* of test statistics of Wilcoxon signed-rank test.

The statistical tests indicate that the proposed method statistically outperforms the other methods in terms of *AUC* and *F-measure* of both minority and majority class, although in case of *G-mean* measure, the proposed method does not seem to outperform *SMOTE* and *ADASYN*. Since we use *AUC* for comparison purpose, it can be inferred that our proposed method is superior to other oversampling methods.

4 Conclusion

In this paper, we propose an oversampling method that adaptively handles between class imbalance and within class imbalance simultaneously. The method identifies the concepts present in the data set using model based clustering and then eliminates the between class and within class imbalance simultaneously by oversampling the sub-clusters where the number of examples to be oversampled is determined based on the complexity of the sub-clusters. The method focuses on improving the test accuracy by adaptively expanding the size of sub-clusters in order to cope with unseen test data. 12 publicly available data sets were analyzed and the results show that the proposed method outperforms the other methods in terms of different performance measures such as *F-measure* of both the majority and minority class and *AUC*.

The work could be extended by testing the performance of the proposed method on highly imbalanced data sets. Further, in our current study, we have expanded the size of clusters uniformly. This could be extended by incorporating the complexity of the surrounding sub-clusters in order to adaptively expand the size of various sub-clusters. This may reduce the possibility of overlapping with other class sub-clusters resulting in increase of classification accuracy.

References

1. Alcalá, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., Herrera, F.: Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *J. Multiple-Valued Log. Soft Comput.* **17**(2–3), 255–287 (2010)
2. Alshomrani, S., Bawakid, A., Shim, S.O., Fernández, A., Herrera, F.: A proposal for evolutionary fuzzy systems using feature weighting: dealing with overlapping in imbalanced datasets. *Knowl.-Based Syst.* **73**, 1–17 (2015)
3. Barua, S., Islam, M.M., Yao, X., Murase, K.: Mwmote-majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Trans. Knowl. Data Eng.* **26**(2), 405–425 (2014)
4. Batista, G.E., Prati, R.C., Monard, M.C.: A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explor. Newsl.* **6**(1), 20–29 (2004)
5. Bradley, A.P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recogn.* **30**(7), 1145–1159 (1997)
6. Burez, J., Van den Poel, D.: Handling class imbalance in customer churn prediction. *Expert Syst. Appl.* **36**(3), 4626–4636 (2009)
7. Ceci, M., Pio, G., Kuzmanovski, V., Džeroski, S.: Semi-supervised multi-view learning for gene network reconstruction. *PLoS ONE* **10**(12), e0144031 (2015)
8. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **16**, 321–357 (2002)
9. Chawla, N.V., Lazarevic, A., Hall, L.O., Bowyer, K.W.: SMOTEBoost: improving prediction of the minority class in boosting. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) *PKDD 2003. LNCS (LNAI)*, vol. 2838, pp. 107–119. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39804-2_12
10. Cleofas-Sánchez, L., García, V., Marqués, A., Sánchez, J.S.: Financial distress prediction using the hybrid associative memory with translation. *Appl. Soft Comput.* **44**, 144–152 (2016)
11. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **7**(Jan), 1–30 (2006)
12. Díez-Pastor, J.F., Rodríguez, J.J., García-Osorio, C.I., Kuncheva, L.I.: Diversity techniques improve the performance of the best imbalance learning ensembles. *Inf. Sci.* **325**, 98–117 (2015)
13. Estabrooks, A., Jo, T., Japkowicz, N.: A multiple resampling method for learning from imbalanced data sets. *Comput. Intell.* **20**(1), 18–36 (2004)
14. Fawcett, T.: ROC graphs: notes and practical considerations for researchers. *Mach. Learn.* **31**(1), 1–38 (2004)
15. Fraley, C., Raftery, A.E.: MCLUST: software for model-based cluster analysis. *J. Classif.* **16**(2), 297–306 (1999)
16. Fraley, C., Raftery, A.E.: Model-based clustering, discriminant analysis, and density estimation. *J. Am. Stat. Assoc.* **97**(458), 611–631 (2002)
17. Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., Herrera, F.: A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.)* **42**(4), 463–484 (2012)
18. García, S., Fernández, A., Luengo, J., Herrera, F.: Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power. *Inf. Sci.* **180**(10), 2044–2064 (2010)

19. Garcia, S., Herrera, F.: An extension on “statistical comparisons of classifiers over multiple data sets” for all pairwise comparisons. *J. Mach. Learn. Res.* **9**(Dec), 2677–2694 (2008)
20. García, V., Mollineda, R.A., Sánchez, J.S.: A bias correction function for classification performance assessment in two-class imbalanced problems. *Knowl.-Based Syst.* **59**, 66–74 (2014)
21. Grant, M., Boyd, S., Ye, Y.: CVX: Matlab software for disciplined convex programming (2008)
22. Guo, H., Viktor, H.L.: Boosting with data generation: improving the classification of hard to learn examples. In: Orchard, B., Yang, C., Ali, M. (eds.) IEA/AIE 2004. LNCS (LNAI), vol. 3029, pp. 1082–1091. Springer, Heidelberg (2004). <https://doi.org/10.1007/978-3-540-24677-0-111>
23. Han, H., Wang, W.-Y., Mao, B.-H.: Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In: Huang, D.-S., Zhang, X.-P., Huang, G.-B. (eds.) ICIC 2005. LNCS, vol. 3644, pp. 878–887. Springer, Heidelberg (2005). https://doi.org/10.1007/11538059_91
24. He, H., Bai, Y., Garcia, E.A., Li, S.: ADASYN: adaptive synthetic sampling approach for imbalanced learning. In: IEEE International Joint Conference on Neural Networks, IJCNN 2008, (IEEE World Congress on Computational Intelligence), pp. 1322–1328. IEEE (2008)
25. He, H., Garcia, E.A.: Learning from imbalanced data. *IEEE Trans. Knowl. Data Eng.* **21**(9), 1263–1284 (2009)
26. Holte, R.C., Acker, L., Porter, B.W., et al.: Concept learning and the problem of small disjuncts. In: IJCAI, vol. 89, pp. 813–818. Citeseer (1989)
27. Huang, J., Ling, C.X.: Using auc and accuracy in evaluating learning algorithms. *IEEE Trans. Knowl. Data Eng.* **17**(3), 299–310 (2005)
28. Japkowicz, N.: Class imbalances: are we focusing on the right issue. In: Workshop on Learning from Imbalanced Data Sets II, vol. 1723, p. 63 (2003)
29. Japkowicz, N., Stephen, S.: The class imbalance problem: a systematic study. *Intell. Data Anal.* **6**(5), 429–449 (2002)
30. Jo, T., Japkowicz, N.: Class imbalances versus small disjuncts. *ACM SIGKDD Explor. Newsl.* **6**(1), 40–49 (2004)
31. Kubat, M., Matwin, S., et al.: Addressing the curse of imbalanced training sets: one-sided selection. In: ICML, vol. 97, Nashville, USA, pp. 179–186 (1997)
32. Lango, M., Stefanowski, J.: Multi-class and feature selection extensions of roughly balanced bagging for imbalanced data. *J. Intell. Inf. Syst.* **50**(1), 97–127 (2018)
33. Lutwak, E., Yang, D., Zhang, G.: L_p John ellipsoids. *Proc. Lond. Math. Soc.* **90**(2), 497–520 (2005)
34. Maldonado, S., López, J.: Imbalanced data classification using second-order cone programming support vector machines. *Pattern Recogn.* **47**(5), 2070–2079 (2014)
35. Piras, L., Giacinto, G.: Synthetic pattern generation for imbalanced learning in image retrieval. *Pattern Recogn. Lett.* **33**(16), 2198–2205 (2012)
36. Prati, R.C., Batista, G.E.A.P.A., Monard, M.C.: Class imbalances *versus* class overlapping: an analysis of a learning system behavior. In: Monroy, R., Arroyo-Figueroa, G., Sucar, L.E., Sossa, H. (eds.) MICAI 2004. LNCS (LNAI), vol. 2972, pp. 312–321. Springer, Heidelberg (2004). <https://doi.org/10.1007/978-3-540-24694-7-32>
37. Provost, F.J., Fawcett, T., et al.: Analysis and visualization of classifier performance: comparison under imprecise class and cost distributions. In: KDD, vol. 97, pp. 43–48 (1997)

38. Richardson, A.: Nonparametric statistics for non-statisticians: a step-by-step approach by Gregory W. Corder, Dale I. Foreman. *Int. Stat. Rev.* **78**(3), 451–452 (2010)
39. Saradhi, V.V., Palshikar, G.K.: Employee churn prediction. *Expert Syst. Appl.* **38**(3), 1999–2006 (2011)
40. Weiss, G.M.: Mining with rarity: a unifying framework. *ACM SIGKDD Explor. Newsl.* **6**(1), 7–19 (2004)
41. Yang, C.Y., Yang, J.S., Wang, J.J.: Margin calibration in SVM class-imbalanced learning. *Neurocomputing* **73**(1), 397–411 (2009)
42. Yen, S.J., Lee, Y.S.: Cluster-based under-sampling approaches for imbalanced data distributions. *Expert Syst. Appl.* **36**(3), 5718–5727 (2009)
43. Yu, D.J., Hu, J., Tang, Z.M., Shen, H.B., Yang, J., Yang, J.Y.: Improving protein-atp binding residues prediction by boosting svms with random under-sampling. *Neurocomputing* **104**, 180–190 (2013)



From Measurements to Knowledge - Online Quality Monitoring and Smart Manufacturing

Satu Tamminen¹(✉), Henna Tiensuu¹, Eija Ferreira¹, Heli Helaakoski²,
Vesa Kyllönen², Juha Jokisaari³, and Esa Puukko⁴

¹ University of Oulu/BISG, Oulu, Finland

satu.tamminen@oulu.fi

² VTT, Oulu, Finland

³ SSAB Europe, Raahe, Finland

⁴ Outokumpu Stainless, Tornio, Finland

<http://www.oulu.fi/bisg>

Abstract. The purpose of this study was to develop an innovative supervisor system to assist the operators in an industrial manufacturing process to help discover new alternative solutions for improving both the products and the manufacturing process.

This paper presents a solution for integrating different types of statistical modelling methods for a usable industrial application in quality monitoring. The two case studies demonstrating the usability of the tool were selected from a steel industry with different needs for knowledge presentation. The usability of the quality monitoring tool was tested in both case studies, both offline and online.

Keywords: Data mining · Smart manufacturing · Online monitoring
Quality prediction · Knowledge representation · Machine learning

1 Introduction

Knowledge can be considered to be the most valuable asset of a manufacturing enterprise, when defining itself in the market and competing with others. The competitiveness of today's industry is built on quality management, delivery reliability and resource efficiency, which are dependent on the effective usage of the data collected from all possible sources. The risk is that the operators with the limited capacity to process the incessant information flow miss the essential knowledge within the data. Recent advances in statistical modelling, machine learning and IT technologies create new opportunities to utilize the industrial data efficiently and to distribute the refined knowledge to end users in right time and convenient format.

Manufacturing has benefited from the field of data mining in several areas, including engineering design, manufacturing systems, decision support systems,

shop floor control and layout, fault detection, quality improvement, maintenance, and customer relationship management [1]. While the amount of data expands rapidly, there is a need for automated and intelligent tools for data mining. Statistical regression and classification methods have been utilized for steel plate monitoring [2]. Decision support systems (DSS), for example, become intelligent when combined with AI tools such as fuzzy logic, case-based reasoning, evolutionary computing, artificial neural networks (ANN), and intelligent agents [3, 4].

Knowledge engineering and data mining have enabled the development of new types of manufacturing systems. Future manufacturing is able to adapt to demands of agile manufacturing, including a rapid response to changing customer requirements, concurrent design and engineering, lower cost of small volume production, outsourcing of supply, distributed manufacturing, just-in-time delivery, real-time planning and scheduling, increased demands for precision and quality, reduced tolerance for errors, in-process measurements and feedback control [5]. Smart manufacturing will bring solutions to existing challenges, but the current industry utilizes generally the information from its environment and in best cases only the first level of knowledge (Fig. 1). The progress in industrial data utilization is enabled with novel intelligent data processing methods.

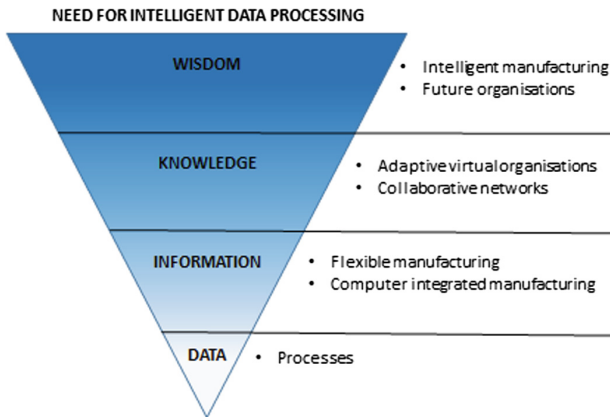


Fig. 1. The evolution of data to knowledge requires novel methods for intelligent data processing that enable the shift to smart manufacturing.

Bi *et al.* state that every major shifting of manufacturing paradigm has been supported by the advancement of IT. Internet of Things (IoT) may change the operation and role of many existing industrial systems in manufacturing. The integration of sensors, RFID tags and communication technologies into the production facilities enables the cooperation and communication between different physical objects and devices [6]. One of the technical challenges in IoT research is the question how to integrate IoT with existing IT systems. When the massive amount of real-time data flow is to be analysed, currently strong big data analytics skills are needed from the end user [7]. However, the employment of

experts concentrate on the core area of the industry, which in its turn, generate a demand for intelligent tools for decision support.

Information presentation is a complex task in manufacturing, as the amount of quality parameters that need to be linked with even a larger number of process parameters is difficult to process with capabilities of a human being. Akram *et al.* show how statistical process control (SPC) and automatic process control (APC) can be integrated for process monitoring and adjustment [8]. Statistical models bring wider possibility to produce information with their capability to predict the future outcome, which enables the process and production planning. The challenge is how to enable the communication between people, how to get information that they need from the process or the product, if the information transfer is enabled between the work posts or manufacturing facilities, or how to provide information about the malfunction or decreased quality of the products. The information should be presented clearly, solutions for the problem, also warnings if automatic corrective actions are enabled. As a whole, the information chain should be supported with a tool that enables the knowledge based conversation within the company.

When product quality improvement is pursued, Kano and Nakagawa suggest that the process monitoring system should have at least the following functions: it should be able to predict product quality from operating conditions, to derive better operating conditions that can improve the product quality, and to detect faults or malfunctions for preventing undesirable operation. They have used soft sensors for quality prediction, optimization for operating conditions improvement, and multivariate statistical process control (MSPC) for fault detection in steel industry application [9]. From these objectives, the derivation of better operating conditions may be the most difficult one to reach; even the definition for better conditions can be challenging to draw, as the conditions are often a compromise of least harmful and cost efficient practices.

In this article, we propose a method for online quality monitoring during a manufacturing process with two application cases in steel industry. Our tool links together the statistical models for prediction of quality properties based on the process settings and variables, and presents the results with easily interpretable visualisations. This paper is organized as follows. Section 2 describes the requirements and specifications for online quality monitoring tool for industrial use. Section 3 presents the choice of the modelling method for quality monitoring purposes. The quality prediction based tools for decision support in two case studies are presented in Sect. 4. Section 5 concludes the quality monitoring development.

2 Developing a Quality Monitoring Tool for Industrial Use

2.1 The Domain Requirements and Requests

We launched the development of the quality monitoring tool (QMT) by surveying the requirements set by business, end users and IT environment of the

process industry. We arranged a series of workshops with participating companies, development partners and other stakeholders.

When specifying the requirements of the QMT, we selected the key user groups for the first development step. This way, the tool could be demonstrated with users having different needs. The selection had the highest impact on the user interface and information presentation.

Technical specifications of the quality tool were stable and reliable applications, performance, maintainability, scalability: adding new modules, features, methods or algorithms should be easy, security, authentication, recoverability, standards and tools (programming languages) and accessibility (web application) are also needed.

The QMT prototype is illustrated in Fig. 2. The transfer of the information from the manufacturing process to the end users is presented in the following four steps that are (1) data acquisition, (2) data storage, (3) information analysis and (4) information delivery. In most advanced visualizations in our tool, the information has been refined to knowledge with automatic interpretation of the results.

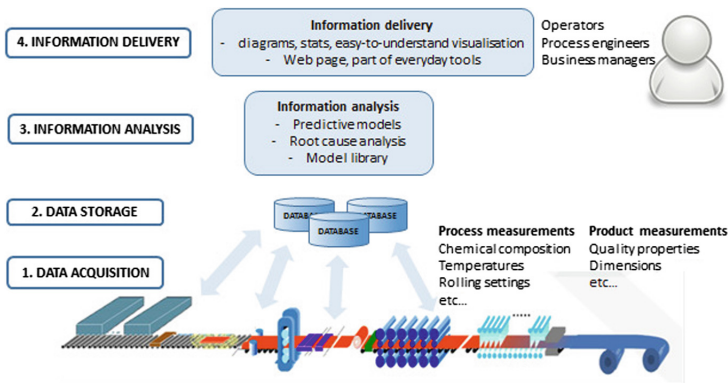


Fig. 2. The prototype of QMT.

2.2 The Specifications for the Tool

The quality information in the QMT is based on the statistical prediction models implemented in R language and equations and rules implemented in C++Mathematical ExpressionToolkit Library (ExprTk). R is a free and open source language for statistical computing. R is integrated into QMT with RServe module, which allows other programs to use facilities of R. R scripts can be written standalone and integration into QMT is straightforward. ExprTk is a mathematical expression parsing and evaluation engine. It is integrated into QMT by including it directly to the source code.

QMT server side is implemented in C++ language. Server side functionality of QMT includes data access and integration into models. Online data access

for QMT is accomplished by reading data from a database. Typically, selected database views are created for accessing data from a database. Some data pre-processing is needed before data can be used for model calculation. For example, a valid range for all model input variables has been defined, and if these limits were violated, model result may not be reliable which is shown with a question mark in the QMT user interface.

The QMT user interface is web based, and in typical use, it provides an overview of process quality as a starting point. In the quality overview, colour coded bars present the quality status of different process phases for each product during the selected time span. Typically, red colour indicates process failure or malfunction, yellow a warning for a process failure and green normal operation. Additionally, white colour indicates that quality information could not be calculated for some reason. Figure 3 illustrates a screen shot from the user interface. The overview to the process shows the predicted quality based on several quality models at different process steps. It hides mathematical models and all process variables from which the quality information is composed. The user can define the relevant quality models to be presented, and if any specific product looks interesting, the tool provides a possibility to analyse it further just by clicking the corresponding bar. Naturally, different user groups require different kinds of views to QMT based on their needs.

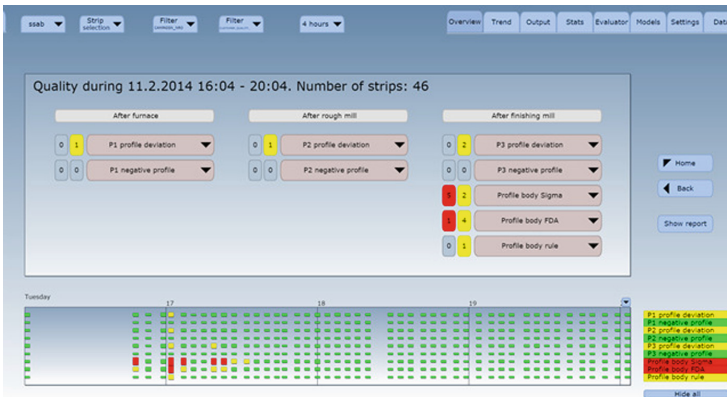


Fig. 3. The user interface of QMT. (Color figure online)

3 Statistical Quality Models

In industrial applications, high nonlinearity and many interactions between process settings challenge the performance of the models. Furthermore, the information about the relations between the predicted variable and the explanatory variables should be available, as for the user, the prediction itself is as valuable as the information about the effects of the process variables to the predicted quality

property. With an online system, the functionality of the tool would suffer, if the models were not capable of processing observations with missing data.

During the last two decades, neural networks have been a popular method for modelling data with complex relations between variables [10–12]. Lately, ensemble algorithms have risen to challenge them with equal accuracy, faster learning, tendency to reduce bias and variance, and they are more likely to over-fit. Seni and Elder state that ensemble methods have been called the most influential development in data mining and machine learning in the past decade [13]. Gradient boosting machines are a family of powerful machine learning techniques that has been successfully applied to a wide range of practical applications [14]. Boosted regression trees are capable of handling different types of predictors and accommodating missing data, there is no need for prior transformation of variables, they can fit complex nonlinear relationships, and automatically handle interactions between predictors [15]. For QMT, the generalized boosted regression models (GBM) were selected, and details of this model can be found in [16]. Juutilainen *et al.* presents in detail how to build models for rejection probability calculation in industrial applications [17].

4 Quality Monitoring in Manufacturing

Two case studies from steel industry were selected to demonstrate the use of the QMT. In case 1, a typical end user is a process engineer with an interest in detailed information about the process and with a need to find root causes for decreased quality. In case 2, a typical end user is an operator with a need for simple and easy-to-interpret presentations about the possibilities of how to improve the quality online.

4.1 Case 1: Strip Profile

A steel strip profile is a quality property for which the product development and the customer set a target value. This information is also essential for the following process steps; especially a negative profile can be very harmful during cold rolling. The target for profile locates typically between 0.03–0.08 mm. Because during the rolling schedule, the target can vary from product to product and strip to strip, profile adaptation is not possible, it is more difficult to hit the target every time. With prediction models, it is possible to design products that more likely fulfil the requirements, as well as to find root causes for the failure. In our QMT, the user can select between the profile and the deviation from the target profile models, depending on the needs.

A typical user could be a process engineer, who wants to learn more about the process and improve it by designing new process practices or product types. The user would expect to have the following outputs that would assist him/her in decision making:

- colour-coded predicted quality during a selected time span for each product

- for a selected product, details about the related process parameters
- information if the model is extrapolating, e.g. some parameter values exceed the training data, and thus, the prediction may be less reliable
- details and visual information about the parameters in the model; what are the most important factors affecting the quality and how do they affect it
- if the product is predicted to have lower quality, how does it differ from the good ones and what could be done differently.

The information flow can get easily overwhelming, and the customization of the result presentation becomes crucial. It is important that the user can find the preferred analysis tools easily and the automated interpretation of the results is provided to speed up the decision making.

By observing the quality prediction model, the user can learn more about the quality property itself and how different process parameters affect it. The strength of the influence for each variable in the model correlates with the actual impact on the quality, and when the quality needs to be improved, the strongest variables are the first candidates to be considered. Figure 4 presents the relative influence of the variables in the profile deviation model. For example, process factors that relate to strength of the steel have a high impact on the profile deviation risk.

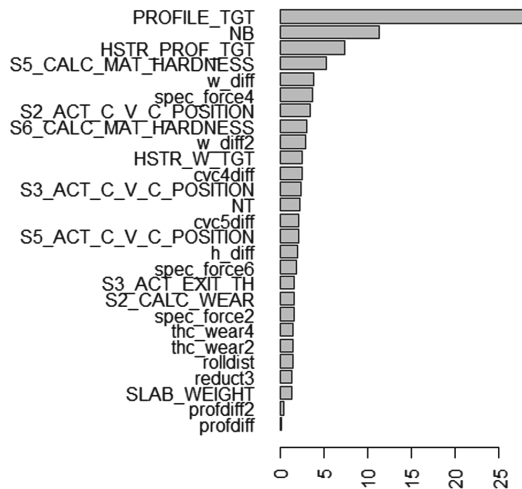


Fig. 4. The visualized variable importance in the GBM model for quality prediction.

The GBM model enables the visual inspection of the effect of each variable in the model. Thus, the user can learn to understand the manufacturing process better. Figure 5 presents two variables from a profile deviation model. The desired value for the property would be zero, and it can be seen that changes in strip width from product to product will increase the risk of profile

deviation to both directions, whereas the large positive values of roll position ($S2_ACT_C_V_C_POSITION$) will increase the risk of negative profile deviation.

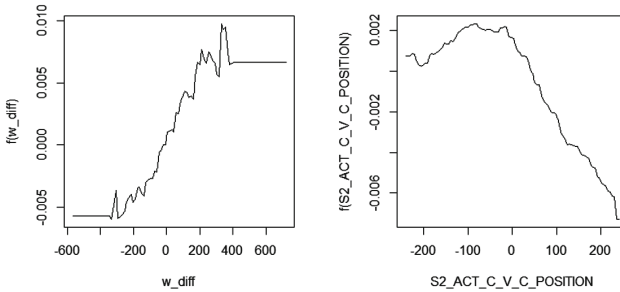


Fig. 5. The visualized effects of two variables in the GBM model for profile deviation prediction.

While analysing the prediction results of a product with a risk of a failure, the user typically asks what the difference between the observed one and good products is. Visually the difference can be effectively presented with parallel coordinates [18]. The comparison is not simple though, because, the determination of a similar good products is a complex procedure. There is not only one way of making each product, but it is always a compromise forced by the status of the production line. Furthermore, there can be hundreds of different products with small modifications depending on the customer. In this application, the weight and height of a strip were used as similarity measures when fetching the best products from a pool of good products that can be used as examples of good production practices. Figure 6 presents two examples of products with negative predicted deviation from the profile target (black) and their comparison with similar good products (green). In the upper case, the observed product seems to have a bit higher value for parameters 2 and 7, but no clear candidates for quality improvement cannot be determined. In the lower case, variables 2, 3, 4 and 7 have a significant difference to the good products, and thus, the user will learn that those settings possess a higher risk for failure.

The comparison can be done also by calculating the distances between the curves. This way, a threshold can be set for showing high enough deviations from the good products. In Fig. 7, the distances of an observed product are visually compared with the good ones. The customisation of the QMT has been made possible by offering several visualisation tool choices for the user.

4.2 Case 2: Strip Roughness

The central roughness of a stainless steel strip is a defect that appear after cold-rolling and surface treatments. The tendency to suffer from this defect type depends on the chemical composition and mechanical properties of the product,

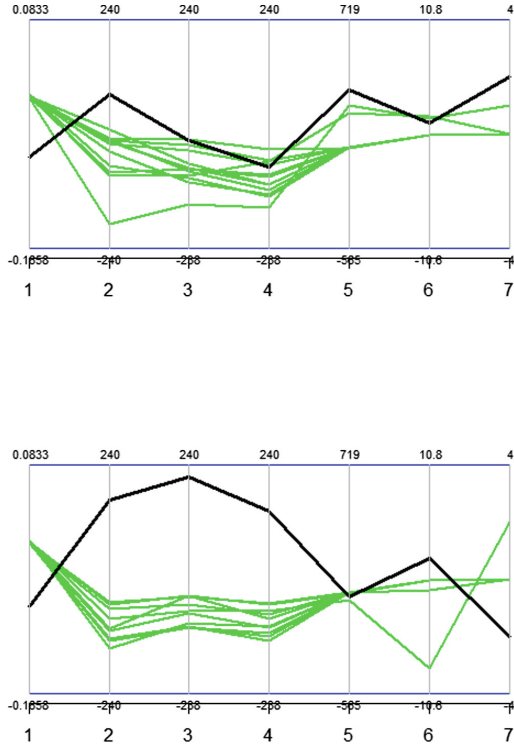


Fig. 6. The parallel coordinates visualize the difference between the good products (green) and the observed product (black) having an increased predicted risk for a failure. (Color figure online)



Fig. 7. The deviations of the model variables from the good products for a selected observation.

but also cold-rolling process parameters have a high impact on the surface. The QMT provides the user an easy to follow overall view to the process, and the user gets simple suggestions for improving the process, if an increased risk of defect occurred.

A typical user could be a process operator, who needs to concentrate on various information sources simultaneously. The presentation of the predicted quality have to be simple and it should support decision making when there is a limited time to react. The user would expect to have the following outputs that would assist him/her in decision making:

- colour-coded predicted quality during a selected time span for each product
- clear visualization of recommended actions for a product with defect risk.

It is important that only relevant information is presented, or the user might start to neglect it. The chemical composition may have a high influence on the product quality, but at this point, the process operator have no possibility to modify it, and thus, the information is meaningless for the user. Instead, the process engineer is responsible for the improvement of the whole manufacturing process.

Figure 8 presents the information provided for the process operator, when the observed product has a risk of surface defect. It is easy to select which parameter should be adjusted, when no distracting information is present.

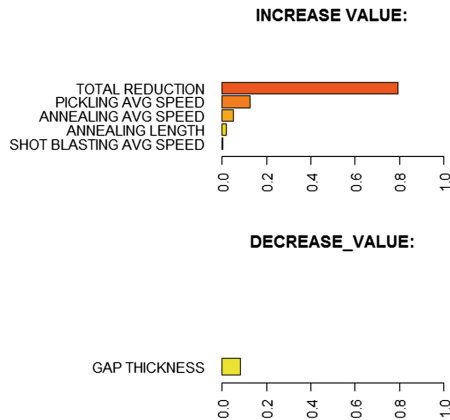


Fig. 8. Recommendations for process improvement.

5 Conclusion and Perspectives

This paper presented an online quality monitoring tool for information acquisition and sharing in manufacturing. The web based tool provides decision support for users in different roles in manufacturing. Furthermore, the user can find root-causes for the reduced quality and learn how to improve the process.

Statistical quality models predict the quality of each product during the manufacturing, and the results are colour-coded to easily interpreted visual presentations. When the user notices a deviant product or a period of defected products, it is easy to fetch more information about the product by selecting suitable actions. The provided visualizations will help to understand the model and factors that affect the prediction, and thus, the predicted quality as well. More advanced methods link the observed products with successful similar products and highlight the differences in production. It can also recommend actions for quality improvement.

The QMT is having an online test period at both participating factories. The user feedback will provide us valuable information for further development of the tool. New user groups with different needs for information presentation will be included in the tool later. In its current version, the selected product can be compared with good ones fetched from a saved data set that has a large presentation of different product types. Later, the dynamicity will be improved by allowing the QMT to fetch an up to date comparison data from the online data base. As a result, it will be faster to find process settings that may be causing quality issues in constantly changing environment.

References

1. Harding, J., Shahbaz, M., Srinivas, Kusiak, A.: Data mining in manufacturing: a review. *J. Manuf. Sci. Eng.* **128**, 969–976 (2006)
2. Siirtola, P., Tamminen, S., Ferreira, E., Tiensuu, H., Prokkola, E., Rönning, J.: Automatic recognition of steel plate side edge shape using classification and regression models. In: *Proceedings of the 9th Eurosim Congress on Modelling and Simulation (EUROSIM 2016)* (2016)
3. Phillips-Wren, G.: Intelligent decision support systems. In: *Multicriteria Decision Aid and Artificial Intelligence*. Wiley, Chichester (2013)
4. Logunova, O., Matsko, I., Posohov, I., Luk'ynov, S.: Automatic system for intelligent support of continuous cast billet production control processes. *Int. J. Adv. Manuf. Technol.* **74**, 1407–1418 (2014)
5. Dumitrache, I., Caramihai, S.: The intelligent manufacturing paradigm in knowledge society. In: *Knowledge Management. InTech*, pp. 36–56 (2010)
6. Bi, Z., Xu, L., Wang, C.: Internet of things for enterprise systems of modern manufacturing. *IEEE Trans. Ind. Inf.* **10**(2), 1537–1546 (2014)
7. Xu, L., He, W., Li, S.: Internet of things in industries: a survey. *IEEE Trans. Ind. Inf.* **10**(4), 2233–2243 (2014)
8. Akram, M., Saif, A.W., Rahim, M.: Quality monitoring and process adjustment by integrating SPC and APC: a review. *Int. J. Ind. Syst. Eng.* **11**(4), 375–405 (2012)
9. Kano, M., Nakagawa, Y.: Data-based process monitoring, process control, and quality improvement: recent developments and applications in steel industry. *Comput. Chem. Eng.* **32**(1–2), 12–24 (2008)
10. Bhadesia, H.: Neural networks in materials science. *ISIJ Int.* **39**(10), 966–979 (1999)
11. Tamminen, S., Juutilainen, I., Rönning, J.: Quantile regression model for impact toughness estimation. In: Perner, P. (ed.) *ICDM 2010. LNCS (LNAI)*, vol. 6171, pp. 263–276. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14400-4_21

12. Tamminen, S., Juutilainen, I., Röning, J.: Exceedance probability estimation for quality test consisting of multiple measurements. *Expert Syst. Appl.* **40**, 4577–4584 (2013)
13. Seni, G., Elder, J.: Ensemble methods in data mining: improving accuracy through combining predictions. In: *Synthesis Lectures on Data Mining and Knowledge Discovery*. Morgan & Claypool, USA (2010)
14. Natekin, A., Knoll, A.: Gradient boosting machines, a tutorial. *Front. Neurobot.* **7** (2013)
15. Elith, J., Leathwick, J., Hastie, T.: A working guide to boosted regression trees. *J. Anim. Ecol.* **77**, 802–813 (2008)
16. Friedman, J.: Stochastic gradient boosting. *Comput. Stat. Data Anal.* **19**, 367–378 (2002)
17. Juutilainen, I., Tamminen, S., Röning, J.: A tutorial to developing statistical models for predictiong disqualification probability. In: *Computational Methods for Optimizing Manufacturing Technology, Models and Techniques*, pp. 368–399. IGI Global, USA (2012)
18. Inselberg, A.: Visual data mining with parallel coordinates. *Comput. Stat.* **13**(1), 47–63 (1998)



Mining Sequential Correlation with a New Measure

Mohammad Fahim Arefin, Maliha Tashfia Islam,
and Chowdhury Farhan Ahmed^(✉)

Department of Computer Science and Engineering,
University of Dhaka, Dhaka, Bangladesh
f.arefin8@gmail.com, maliha.tashfia@gmail.com, farhan@du.ac.bd

Abstract. Being one of the most useful fields of data mining, sequential pattern mining is a very popular and much researched domain. However, simply pattern mining is often not enough to understand the intricate relationships that exist between data objects or items. A correlation measure can uplift the task of mining interesting information that is useful to the end user. In this paper, we propose a new correlation measure, *SequentialCorrelation*, for sequential patterns. Along with that, we propose a complete method called *SCMine* and design its efficient trie-based implementation. We use the measure to define a one or two way relationship between data objects and subsequently classify patterns into two subsets based on order dependency. Our performance study shows that a number of insignificant patterns can be pruned and it can give valuable insight into the datasets. *SequentialCorrelation* along with *SCMine* can be very useful in many real life applications, especially because conventional correlation measures are not applicable in sequential datasets.

Keywords: Sequential pattern · Sequential correlation
Data mining · Sequential dependency

1 Introduction

Data mining is a field of science that deals with obtaining information (possibly unknown, interesting) from a huge amount of raw, unstructured data or repositories. One of the recently popular fields of data mining is sequential pattern mining. Sequential pattern mining [5] is quite similar to the classic data mining domain of frequent itemset mining. The main difference between the two is that, the order of items or data objects are not relevant in frequent itemset mining whereas sequential pattern mining specifically deals with data sequences where items are ordered. Sequential pattern mining methods are popularly used to identify patterns which are usually used in making recommendation systems, text predictions, improving system usability, making informative product choice decisions.

Many a times, even mining the frequent patterns or sequences are not enough. We would get a huge amount of patterns in lower support thresholds and only

the obvious information from high thresholds. Correlation analysis is a useful tool here. Correlation analysis basically means finding out or measuring the strength of relationship among items, itemsets or data objects. The main motivation behind our work lies in the fact that there are not many widely known or standard correlation measures for sequential patterns.

For example, let's suppose laptops and portable hard drives are frequently bought from a tech shop. Furthermore, there are 8 occurrences of Laptop \implies Hard Drive and 2 occurrences of Hard Drive \implies Laptop. In the total dataset there are 10 occurrences of each. In lower support thresholds, both these patterns are frequent but obviously we can decipher more about their relationship from the frequencies. There's a 80% possibility that a laptop purchase will be followed by a purchase of hard drive, which means hard drives are generally bought after laptops.

Because we are working with sequential patterns, it is important that we retain information about the order in which they appeared while mining. If the sale of hard drives is found to be followed by the sale of laptop to a significant degree, this can be used in the real life application to boost sales or improve service. Otherwise if the order is not significant enough, advertising can be done in any form irrespective of order.

Our main contributions have been finding a null invariant correlation measure for sequential patterns and constructing a complete method of using this measure, while keeping in mind the overhead for correlation analysis and performance benefits.

In the next section, some overview of previous works related to our field of application has been given. Section 3 contains the approach and algorithm with a short demonstration towards the end. Section 4 discusses the performance study and results obtained from it. Finally, we conclude with a small discussion about the future scope of our proposed methodology in Sect. 5.

2 Related Work

There are multiple sequential pattern mining algorithms. The most widely used one is **PrefixSpan** [1]. Given a sequence database and the minimum support threshold, PrefixSpan finds the complete set of sequential patterns in the database. It adopts a divide-and-conquer, pattern-growth principle by recursively projecting sequence databases into a set of smaller projected databases based on the current sequential pattern(s). Projected database is a collection of suffixes with respect to a specific prefix. Then sequential patterns are grown in each projected databases by exploring only locally frequent fragments. Physical projection of a sequence can also be replaced by registering a sequence identifier.

PSBSPan [7] is an algorithm based on pattern growth methodology for mining frequent correlated sequences. The basic idea is that, a frequent sequence is correlated if the items in the sequence are more probable to appearing together as a sequence rather than appearing separately. Using this ratio of probability, a prefix and suffix upperbound can be calculated for each sequence. The algorithm

works basically like PrefixSpan, the only addition being that a frequent sequence is only selected at each step if its prefix/suffix upperbound crosses a correlation threshold.

S2MP [10] is a similarity measure to compute the similarity of sequential patterns, which takes the characteristics and the semantics of sequential patterns into account. It considers the position of items as well as the number of common and non-common items in a sequence. A mapping score is given based on the resemblance of two sequences and an order score is given based the order and position of items in the sequences. Finally, S2MP is an aggregation of the two scores.

The **CMRules** [3] is an association rule mining algorithm, which is based on the fact if we ignore or remove the time information of a sequence database, we obtain a transaction database. The algorithm ignores sequential information and mines the transaction database based on a minimum support and confidence threshold and generates the association rules. After that, sequential support and confidence of each rule is calculated by the sequential database and rules that do not meet the thresholds are eliminated. Since it uses general association rule mining, the execution time for CMRules grows proportionally to the number of association rules.

3 Our Proposed Approach

In this section, we will discuss:

- Our proposed measure *SequentialCorrelation*
- A complete algorithm for pattern categorization, *SCMine*, along with its pseudo-code
- A suitable example for better understanding.

3.1 Terminologies

Itemset and Sequence: An itemset is denoted as (x_1, x_2, \dots, x_k) , where x_k is an item. A **sequence** is an ordered list of itemsets. A sequence S is denoted by $\langle s_1s_2, \dots, s_l \rangle$, where s_j is an itemset or an element of the sequence.

Reverse Sequence: If a multi-itemset sequence is frequent, all of its sub patterns will be frequent also. So we consider each of the itemset to be atomic because an itemset is present in frequent pattern when the items in the itemset occur frequently, so it is significant as a whole.

So if $S = \langle A(AB)CD \rangle$, then reverse sequence of S , $S' = \langle DC(AB)A \rangle$.

Order Dependent or One Way Sequences: An order dependent sequence is formally defined as a sequence whose items maintain a dominant order of appearance in the sequential database. If A and B are items or itemsets in a

sequential dataset then the sequence AB will be order dependent if the event, e_A , containing A mostly comes before the event, e_B , which contains B.

It is also denoted as a one way sequence because AB will be frequent in the dataset while BA falls below the user defined support threshold i.e. the reverse sequence is infrequent.

Order Independent Sequences: An order independent sequence is defined as a sequence whose items have flexible order of appearance in the sequential database. For example, the sequence AB will be order independent if the event, e_A , containing A and the event e_B , which contains B have no dominant order. In other words, e_A can occur before and after e_B in similar frequency in the total dataset.

SequentialCorrelation: Our measure, *SequentialCorrelation*, works by taking the ratio of a certain pattern and its reverse with respect to their total occurrence. This gives us an estimate about the percentage or amount of patterns that follow a certain direction. We do not consider the total number of transactions because we wanted the measure to be null invariant.

Let $F(A)$ be such a function that generates the frequency or total count of sequence A in the dataset. Here, A can be a sequence of itemsets. A' will be the reverse sequence. For Example, if $A = \langle A(AB)CD \rangle$, then $A' = \langle DC(AB)A \rangle$. So the Sequential Correlation becomes:

$$\text{SequentialCorrelation}, SC(A) = \frac{F(A) - F(A')}{F(A) + F(A')} \quad (1)$$

The sequential correlation score will always have a value in the range $[-1, 1]$ as all the itemsets in A will be frequent items or itemsets. $|SC| = 0$ will mean items are independent and $|SC| = 1$ will mean items have a very strong order dependency. A negative score represents that the reverse sequence is more significant. The score is also null-invariant as it doesn't take null transactions into consideration.

SequentialCorrelation Threshold (SC. Threshold): SequentialCorrelation Threshold denotes the tolerance or benchmark level for order dependency. Using a high threshold means that highly order dependent sequences are taken as interesting. A low threshold means that patterns or sequences can have flexibility in ordering items. It is denoted as *SC.Threshold*. Like support-confidence threshold, there is no universal *SC.Threshold* value that is used. The threshold depends on the type of database, the data it carries and the particular objectives of the user.

3.2 Trie Architecture

For the purpose of efficient storing and retrieving of patterns, the trie data structure has been used. It is also known as prefix tree. The trie is efficient in

memory usage because a lot of patterns or sequences will have the same prefix or a partially similar prefix. It is essentially a tree with each node representing an element of the sequence. Each prefix is also stored only once.

Suppose, the dataset contains the sequences $\{ABC:2, AD:3, BD:2\}$. The trie will have a root which contains links to the start of each sequence. The endpoint flags if the sequence up to that node is actually found in the dataset or it is just an intermediate node. The trie will look similar to Fig. 1.

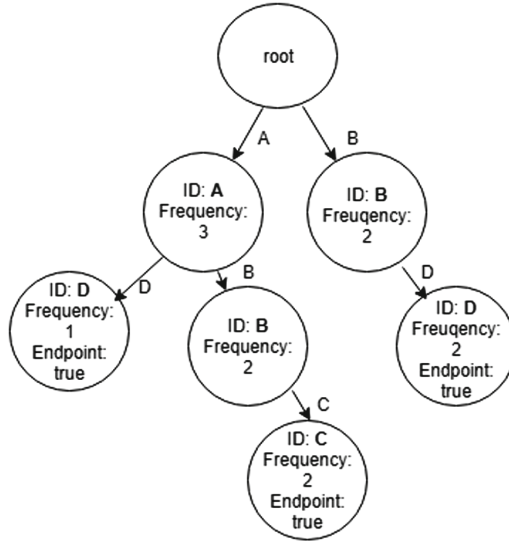


Fig. 1. Structure of the trie

3.3 Algorithm: *SCMine*

Given a sequential database and the set of frequent sequences based on a user-defined support threshold, we need to find the sets of order dependent sequences and order independent sequences by using *SequentialCorrelation* measure. The *SC.Threshold* is also user-defined.

Step by Step Procedure: The prerequisite of our algorithm is a frequent pattern mining step. *SCMine* assumes that any acceptable sequential pattern mining algorithm has been used to generate the list of frequent sequential patterns and their frequency count. Then the following steps are taken:

- **Build trie from frequent patterns:** For each frequent pattern, the pattern along with its frequency or count is stored into the trie. At the end of this step, the trie will contain the complete set of frequent patterns.

- **Calculate *SequentialCorrelation* score:** At this step, the correlation score for each frequent pattern is calculated from Eq. (1). To calculate the score, the reverse pattern frequency is also looked up from the trie.
- **Categorize patterns:** After calculating the *SequentialCorrelation* score for each pattern, they can now be categorized into order dependent or order independent sequences based on *SC.Threshold*. If the score of a pattern crosses the threshold it is categorized as order dependent.

3.4 Pseudocode

The *SCMine* algorithm starts from line 4, by calling the *SCMINE()* procedure with the list of frequent patterns or sequences generated by *PrefixSpan* and the user defined *SC.Threshold*. In line 5–7, the trie is prepared with all the frequent patterns and their frequencies. From line 8, each pattern is processed. Line 11 provides the first condition of classification or partitioning. In line 14, the procedure *SEQUENTIALCORRELATION()* is called. The procedure is defined in line 1–3. It performs a lookup on the trie to get the frequencies needed and uses simple mathematics to calculate the return value. In line 15 and 17, the score is compared and based on the comparison, different partitioning decisions

Algorithm 1. Calculating sequential correlation

```

1: procedure SEQUENTIALCORRELATION(A)
2:    $F \leftarrow$  frequency of  $A$ 
3:    $F' \leftarrow$  frequency of  $A'$ 
    $score := \frac{F-F'}{F+F'}$ 
   return  $score$ 
4: procedure SCMINE(LIST OF ALL FREQUENT PATTERNS, SC.THRESHOLD)
5:   Trie  $T \implies \emptyset$ 
6:   for (all frequent patterns) do
7:     Insert into  $T$ 
8:   for (all frequent patterns) do
9:      $A \leftarrow$  pattern  $S$ 
10:     $A' \leftarrow$  reverse of pattern  $S$ 
11:    if  $A'$  is not frequent
12:      then  $A$  is an order dependent sequence
13:    else begin
14:       $score :=$  SEQUENTIALCORRELATION( $A$ )
15:      if ( $|score| < SC.Threshold$ )
16:        then  $A$  and  $A'$  are order independent sequences
17:      else if ( $score \geq 0$  and  $|score| \geq SC.Threshold$ )
18:        then  $A$  is an order dependent sequence and dominant in order
19:      else
20:         $A'$  is an order dependent sequence and dominant in order
21:    end
22:  end
23:

```

are taken. If the absolute value of the score is less than the threshold, then it is put into the order independent subset.

3.5 Demonstration

We use a minimized dataset containing 10 distinct items and 10 sequences for demonstration purpose here (Table 1):

Table 1. Sequences of the custom dataset (minimized)

Code	Item	Sequences
C	Cooler	L M S E R
D	Desktop	D U S L E
E	Earphone	L M S R E
K	Keyboard	L E S P M
L	Laptop	L M S P E
M	Mouse	L E C S M P
P	Pen Drive	S E R L
R	Router	L S E
S	Smartphone	S L E R P
U	UPS	S E M R K L

Single-Item Sequences: L:10, S:10, E:10, R:5, M:5, P:4, D:1, K:1, C:1, U:1. At 20% minimum support threshold, we get 6 items: L, S, E, R, M, P.

Generating Sequential Patterns: Calculating sequences prefixed with S:

- Set of Suffixes with prefix S in dataset:

$$\langle ER, LE, RE, PM, PE, MP, ERL, E, LERP, EMRKL \rangle$$

From this, we calculate the frequent items in this set E:8, R:5, L:4, P:4, M:3.

- Set of suffixes with prefix SE in dataset:

$$\langle R, RL, RP, MRKL \rangle$$

From this we calculate the frequent items in this set: R:4, L:2.

- Set of suffixes with prefix SER in dataset:

$$\langle L, P, KL \rangle$$

From this we calculate the frequent items in this set: L:2.

- Set of suffixes with prefix SERL in dataset is empty. So this pattern will not get any longer.

The sequential patterns with prefix S are listed in Table 2.

Table 2. Sequential patterns with prefix S

Prefix	Projected (suffix) database	Sequential pattern
<S>	<ER>, <LE>, <RE>, <PM>, <PE>, <MP>, <ERL>, <E>, <LERP>, <EMRKL>	<S>, <SL>, <SE>, <SR>, <SM>, <SK>, <SP>
<SL>	<E>, <ERP>	<SLE>
<SE>	<R>, <RL>, <RP>, <MRKL>	<SEL>, <SER>, <SERL>
<SR>	<E>, <L>, <P>, <KL>	<SRL>
<SM>	<P>, <RKL>	-
<SK>	<L>	-
<SP>	<M>, <E>	-

As this is a recursive process, similarly we generate the projected databases for all other frequent prefixes such as - L, E, R, M, P.

We get 20 sequences with 2 items, 21 sequences with 3 items and 5 sequences with 4 items. Let us calculate sequential correlation for the sequence $\langle E-S \rangle$. Here, $F(E-S) = 2$, $F(S-E) = 8$

$$\text{SequentialCorrelation}, SC(E-S) = \frac{2-8}{2+8} = -0.6$$

As the SC is negative, the dominant order of the items is the reverse order and as the value is higher than 0.5, these two items are moderately dependent and the order of dependency is $S \implies E$.

Similarly for $\langle L-R \rangle$, $F(L-R) = 2$, $F(R-L) = 2$

$$\text{SequentialCorrelation}, SC(L-R) = \frac{2-2}{2+2} = 0$$

As the SC is 0, these two items are completely independent. They don't have any specific order.

Again, for $\langle M-S-E \rangle$, $F(M-S-E) = 3$, $F(E-S-M) = 2$,

$$\text{SequentialCorrelation}, SC(M-S-E) = \frac{3-2}{3+2} = 0.2$$

As the SC is less than 0.5, these items are order independent.

To avoid redundancy, we have only listed the frequent sequences with positive SC score in Table 3 because the reverse of these sequences will have the same value, only in negative magnitude. Based on the results from Table 3, we can notice a number of interesting facts. Some of which are listed in Table 4. Such as, Laptop-Earphone, Laptop-Mouse, Smartphone-Earphone have high sequential correlation scores and show significant sequential dependency. On the other hand, Laptop-Router have a relatively low sequential score and presents the fact that they do not have an well established dependency on order. The overall table reflects situation that is generally found in real life.

Table 3. Sequential correlations for 2 itemset sequences

Itemset sequence (A-B)	Freq(A-B)	Freq(B-A)	Total freq.	SC(A-B)
Laptop - Smartphone	6	4	10	0.2
Laptop - Earphone	8	2	10	0.6
Laptop - Router	3	2	5	0.2
Laptop - Mouse	4	0	4	1
Laptop - Pen Drive	4	0	4	1
Smartphone - Earphone	8	2	10	0.6
Smartphone - Router	5	0	5	1
Smartphone - Mouse	3	3	6	0
Smartphone - Pen Drive	4	0	4	1
Earphone - Router	4	0	4	1
Earphone Mouse	3	3	6	0
Earphone - Pen Drive	3	0	3	1
Mouse - Router	2	0	2	1
Mouse - Pen Drive	2	0	2	1

Table 4. Observation based on *SequentialCorrelation* value

Sequence(A)	SC(A)	Observation	Explanation
Laptop-Smartphone	0.2	$0 \leq SC < 0.5$ & SC positive	Weak sequential dependency, items do not always maintain this order
Earphone-Smartphone	-0.6	$0.5 \leq SC \leq 0.75$ & SC negative	Negative sequential dependency, smartphone is mostly bought before earphone
Laptop-Pendrive	1.0	$ SC > 0.75$ & SC positive	Strong sequential dependency, pen drive is always bought after laptop
Earphone-Mouse	0.0	$ SC = 0$	No sequential dependency. These are completely independent

Similarly we can calculate sequential correlation for larger itemsets. Based on the correlation score we can classify the sequences based on the level of sequential dependency. For example, if we set the *SC.Threshold* at 0.6, the two-length patterns could be classified into two classes- order independent patterns ($SC < 0.6$) and order dependent patterns ($SC \geq 0.6$). Let us consider $X = (A(BC)D)$ and $Y = (D(BC)A)$, if $SC(X) = 0.6$, then it is order dependent and if $SC(Y) = 0.2$, then that is order independent.

4 Evaluation Results

In this section, we present the overall performance of our proposed measure *SequentialCorrelation* across several real life datasets. At first, we present the relationship between the data objects found through *SequentialCorrelation* in different datasets at different minimum support thresholds. We also analyze the effect of *SC.Threshold* in different datasets on the ratio of the number of order dependent and independent patterns. Lastly, we analyze runtime and memory consumption.

4.1 Experimental Environment

We have run our experiments on a computer with 1.70 GHz Intel Core i5 PC with 6 GB RAM, running windows 10 operating system. 9 real datasets were used in our experiment and all the datasets were collected from the UCI Machine Learning Repository but some datasets were preprocessed to suit our requirements of ordered sequence. We compare performance between PrefixSpan and our algorithm *SCMine*, both of which were implemented in Java. We also normalize the comparisons to better illustrate the advantages of *SCMine* and the usefulness of *SequentialCorrelation*. Several performance metrics have been considered to observe the performance of *SCMine*, in comparison to PrefixSpan. They are shortly described below.

4.2 Performance Analysis

Number of Order-Dependent Patterns. Our *SequentialCorrelation* can be used to classify which of the patterns have strong order dependency and which of the patterns do not. The lower the *SC.Threshold*, the higher number of patterns will be classified as order dependent patterns. These are illustrated through the results shown below, in different real life datasets.

Dataset 1: Activities of Daily Living (ADL). This dataset is comprised of the activities of daily living [9]. As the items were actual daily life events like showering, grooming, sleeping, we could clearly understand what the relationship between the objects meant. As expected, most of the sequences from this dataset were order dependent while very few sequences were order independent. For example, people watch TV before sleeping but almost none watches TV after they wake up from sleep. As shown in Fig. 2a, even at *SC.Threshold* = 0.5, almost 90% of the sequences were order dependent. We can observe there are a significant number of patterns with $SC = 0.1$ and 0.2 .

Dataset 2: UK-Retail. This is a transactional data set which contains all the transactions occurring between December 1, 2010 and December 09, 2011 for a UK-based online retail store [2]. When we set the *SC.Threshold* at 0.1, among the patterns found, 50419 patterns were order dependent while others were order independent. So there was almost a fifty-fifty split between order dependent and order independent patterns, which is shown in Fig. 2b.

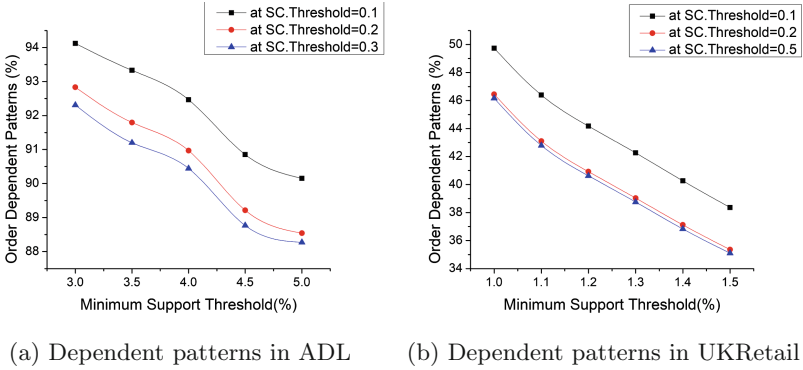


Fig. 2. Order dependent patterns in ADL and UKRetail

Dataset 3: MSNBC. The data comes from the logs for msnbc.com and news-related portions of msn.com for the entire day of September, 28, 1999 [6]. Each sequence in the dataset is the sequence of categories a user visited on that day. When $SC.Threshold$ was set at 0.1, more than 40% of the patterns were classified as order-dependent at different minimum support thresholds. But this ratio drops to below 30% when the $SC.Threshold$ was set at 0.3 or above as observed in Fig. 3a. The patterns which were order dependent were mostly closely related topics.

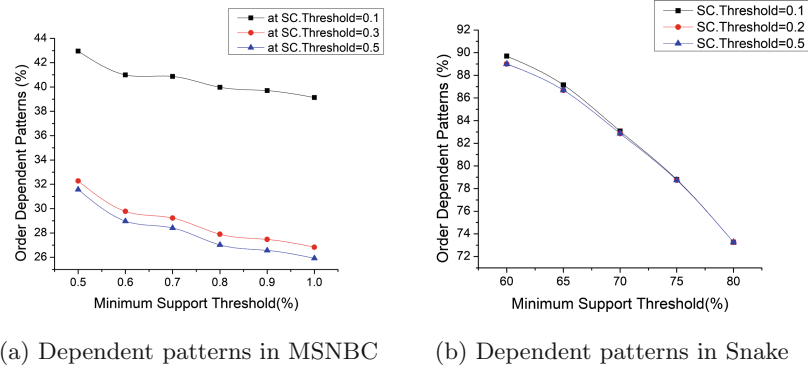


Fig. 3. Order dependent patterns in MSNBC and Snake

Dataset 4: Snake. This dataset is about a family of eukaryotic and viral DNA binding proteins [8]. Is it a very dense dataset, majority of the sequences were order dependent. When minimum support threshold is set at 60%, 89% patterns were order dependent even at $SC.Threshold$ set at 0.5. The ratio doesn't change much depending on the $SC.Threshold$, which means most of the sequences either

occur only in one order or occur in both orders equally which results in a very low SC Score as observed in Fig. 3b.

We had similar results in other Real life datasets such as Bible, Fifa, Leviathan and Sign. Fifa is a dataset of click stream data while Leviathan is a conversion of the novel Leviathan by Thomas Hobbes. The Sign dataset comprises of sign language utterance [4].

4.3 Distribution of Classified Patterns

SCMine classifies the frequent patterns into two classes: order-dependent patterns and order-independent patterns. The ratio of number of order-dependent patterns and number of order-independent patterns differ based on three variables - nature of dataset, minimum support threshold and *SC.Threshold*. To normalize the comparison, we took the minimum support thresholds in such a way that there are around 10,000 frequent sequential patterns with length more than 1. Then we measured the ratio of order dependent and independent patterns in different datasets at different *SC.Threshold*.

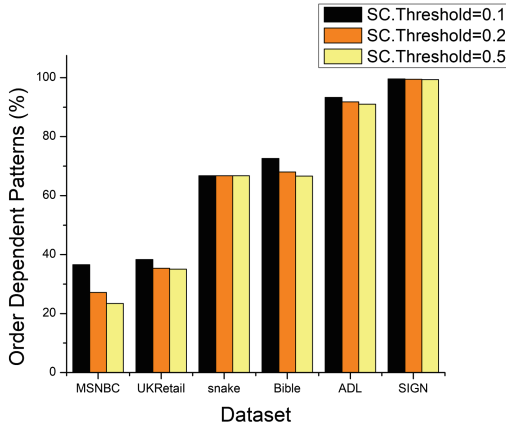


Fig. 4. Percentage of order-dependent patterns in different datasets

Figure 4 illustrate the distribution of classified patterns in different datasets. We can analyze the relationship between the items of a dataset by observing this distribution. For example, there is strict ordering between the items of Sign dataset because almost all the sequences are order-dependent patterns. Similarly there is strong ordering in the ADL dataset because 93% of the patterns are order-dependent. This is a reflection of the real life scenario because the items are daily activities and most of the activities are order dependent.

MSNBC and UKRetail have more order-dependent patterns because the items correspond to webpage categories and retail store items respectively, that's why most of the sequences don't maintain strict ordering between items. Snake is

a biological dataset of protein sequences, so majority of the patterns are order-dependent. Bible is the conversion of the sentences of the bible, so logically majority of the sequences are order-dependent. Some words form different meanings when placed in different order, that’s why there is a significant portion of the sequences which are order-independent.

As we can see in Fig. 4, number of order independent patterns increase as we increase the *SC.Threshold*, this is evident in all the datasets. But the change in the distribution varies across datasets because the distribution is dependent on the nature of the dataset. So different datasets will have different distribution at different support threshold and different *SC.Threshold*.

4.4 Runtime Analysis

For calculating the *SequentialCorrelation*, we took a trie based approach for better efficiency. The frequency of each frequent sequential pattern was stored in a trie. The lookup time required for finding the frequency of an N-length sequence is $O(N)$ and the calculation of *SequentialCorrelation* is done in $O(1)$.

As we mine the *SequentialCorrelation* after mining the frequent patterns using PrefixSpan, we can compare our runtimes with the runtime of running only PrefixSpan to measure the overhead of our algorithm. We present the runtime increase in different datasets at different minimum support thresholds because both of these variables affect the runtime.

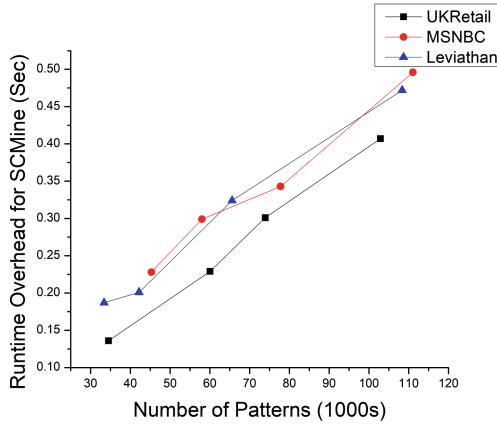


Fig. 5. Runtime comparison across different datasets

There is a close relationship between the number of patterns generated and the runtime because the higher the number of patterns, the more time consumed to mine the *SequentialCorrelation*. Figure 5 represents the relationship between the number of patterns generated and the runtime overhead for *SCMine* in UKRetail, MSNBC and Leviathan dataset. We compare the relationships in

different datasets to highlight the fact that the nature of the dataset has a significant impact on the result because the average length of the patterns also affects the runtime.

4.5 Memory Analysis

The memory overhead for *SCMine* is primarily for the trie structure to store the frequency of the frequent patterns. So the memory usage will increase as the number of frequent patterns increases. The value of the memory overhead depends on two factors- number of frequent patterns generated by the current support threshold and the average length of the frequent patterns, which means the memory overhead will vary depending on the nature of the dataset and the specified minimum support threshold (Fig. 6).

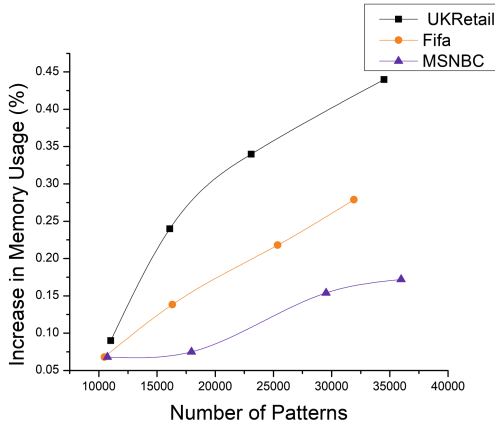


Fig. 6. Memory usage comparison across different datasets

To understand the impact of the datasets on the memory overhead occurred by *SCMine*, we present a comparison of overhead in the datasets MSNBC, UKRetail and Fifa. We compare the relationship between number of patterns generated and memory usage in different datasets. This variation is caused by the length of the patterns because longer patterns take up more space. Another factor was the number of distinct items because the more number of distinct items, the wider the tree will be and so less number of common prefixes.

From the above analysis, *SequentialCorrelation* has been found to work well in multifarious datasets. It can be efficaciously used to trim the number of patterns based on user needs and objectives while adding very limited overhead.

5 Conclusion

Our objective was to define a new correlation measure for sequential patterns and subsequently use the measure to filter interesting rules from the uninterest-

ing ones. We have also analyzed the performance and work-ability of our measure with various datasets and performance metrics. We mainly focused on two-itemset sequences because these are the most primitive ones and are most useful to illustrate the order dependency. Our measure *SequentialCorrelation* evaluates sequential patterns based on order dependency of items and our method *SCMine* shows the complete process of partitioning or classifying patterns. Performance analysis shows that it is satisfactory as it sufficiently reduced frequent patterns to lesser number of interesting patterns, depending on the specified *SC.Threshold*. Time and memory consumption is well within an acceptable limit as trie has been used. However, there is also scope for fine-tuning in order to optimize performance further. This can be developed further to build Associative Classifiers which is a well-known domain of data mining.

References

1. Chand, C., Thakkar, A., Ganatra, A.: Sequential pattern mining: Survey and current research challenges. *Int. J. Soft Comput. Eng.* **2**, 185–193 (2012)
2. 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. Strategy Manag.* **19**, 197–208 (2012)
3. Fournier-Viger, P., Faghihi, U., Nkambou, R., Nguifo, E.M.: CMRules: an efficient algorithm for mining sequential rules common to several sequences. In: FLAIRS Conference. AAAI Press (2010)
4. Fournier-Viger, P., Gomariz, A., Gueniche, T., Soltani, A.: SPMF: a Java open-source pattern mining library. *J. Mach. Learn. Res. (JMLR)* **15** (2014). <http://www.philippe-fournier-viger.com/spmf/>
5. Fournier-Viger, P., Lin, J.C.W., Kiran, R.U., Koh, Y.S.: A survey of sequential pattern mining, vol. 1 (2017)
6. Lichman, M.: UCI machine learning repository (2013). <http://archive.ics.uci.edu/ml/datasets/msnbc.com+anonymous+web+data>
7. Lin, C.X., Ji, M., Danilevsky, M., Han, J.: Efficient mining of correlated sequential patterns based on null hypothesis. In: Proceedings of the 2012 International Workshop on Web-Scale Knowledge Representation, Retrieval and Reasoning, Web-KR 2012. ACM, New York (2012)
8. Liu, T., Huang, J.: Replication protein a and more: single-stranded dna-binding proteins in eukaryotic cells. *Acta Biochimica et Biophysica Sinica* **48**, 665–670 (2016)
9. Morales, J.F., de Toledo, P., Sanchis, A.: Activity recognition using hybrid generative/discriminative models on home environments using binary sensors. *Sensors* **13**, 5460–5477 (2013)
10. Saneifar, H., Bringay, S., Laurent, A., Teisseire, M.: S2MP: similarity measure for sequential patterns. In: Proceedings of the 7th Australasian Data Mining Conference - Volume 87, AusDM 2008. Australian Computer Society Inc., Australia (2008)



A New Approach for Mining Representative Patterns

Abeda Sultana, Hosneara Ahmed, and Chowdhury Farhan Ahmed^(✉)

Department of Computer Science and Engineering,
University of Dhaka, Dhaka, Bangladesh
abida1616@gmail.com, hosneara_17@yahoo.com, farhan@du.ac.bd

Abstract. With the revolution of science and technology, we can accumulate huge amount of data which requires to be manipulated efficiently since the amount of data is expanding hence scarcity of knowledge is also increasing. Therefore analysis for more useful and interesting knowledge is on demand. Representative patterns can be a solution to represent data in a more concise way. Different efficient methods for mining frequent and erasable patterns exist in representative pattern mining field that are regarded as significant. We have proposed a new type of pattern called decaying pattern. These patterns are characterized as those patterns that were frequent for a time being and then decayed with time. These patterns of declining nature can give us the opportunity to analyze reasons behind items' decrease such as extinct animals, finding unsolved accidental news, analysis of buying behavior of customers etc. that require further inspection.

Keywords: Frequent pattern · Erasable pattern
Representative pattern · Pattern tree · Decaying pattern

1 Introduction

Data mining is the process of analyzing large amount data to discover knowledge and finding patterns and relationship among them. By means of data mining we can renovate huge amount of information into useful knowledge. This knowledge is immensely important on various applications and research field. Frequent pattern mining is one of the most significant field of data mining. Another important domain of data mining is mining representative patterns. Different kind of frequent patterns can be formed in itemsets, sequences, episodes and substructures and so on. Representative frequent pattern mining refers to finding precise, distinctive and explicable set of patterns of each class that represent their key characteristics among other classes. Representative patterns represent a dataset and extract the significant knowledge from huge amount of data. This representation can be done in many criteria. Many efficient and noble works are already done on these e.g. some patterns are most significant in the database, that are mined as representative frequent patterns, some patterns are so insignificant that

they are better to be pruned, these type of patterns are mined as representative erasable patterns [7,9,10]. Again maximal, closed, top rank-k, top-k frequent regular [1,2,6] are also developed for representative pattern mining.

There are some patterns which are frequent in a database for some time being, then they are lost. These patterns could be a part of representative frequent pattern set but due to their decaying nature, they fail to get a place there. This type of patterns are ignored but have importance in various research fields. We focus on this decaying representative patterns which can be a means of mining important knowledge from huge amount of data. Erasable closed pattern set consists of the least frequent patterns and subsets of these patterns are also erasable. On the other hand, decaying pattern has two attributes- at first it needs to be highly frequent for a span of time, secondly its frequency will decay and so the resulting pattern becomes infrequent.

1.1 Motivation

With the existing algorithms we can only find the patterns which were most frequent or too scarce. Patterns having both characteristics- that were frequent for a while but have become infrequent with time are not mined yet. This type of patterns are not observed but have importance in various research fields.

Motivating Example

Suppose a large electronics company sells various types of products like laptops, smartphones, smartwatches, tablets etc. Laptops and smartphones are very popular among the buyers of that company. These two products were sold throughout the year. On the other hand, smartwatches are not greatly welcomed by the customers, so the sale of smartwatches remained below the expectation. However, in case of tablet situation was slightly disparate. Notwithstanding, when company released it first, it was a very welcoming product, after there second release the sale suddenly decreased.

After two or three years when a data analyst of that company mined the most remunerative products, he got laptop and smartphone as the number of sale was higher and so the profit. Contrarily, as least profitable product, he got smartwatch. So company will take action for further development of it. As tablets do not fit in any of the two categories, company will never know the problem why the sale of tablets decreased and it will not take any necessary step. If analyst would observe decaying nature in the trade, this tablet will come to light for finding the reason behind consumers' sudden displeasure toward this product.

There are more important applications of decaying patterns. Everyday accidents or unusual occurrences are happening that appear in newspaper and social media so frequently for some days and then perish of being heedless. In most cases, they remain unresolved. Mining those patterns can help correspondents write follow-ups. Again, we can find this pattern in species data of animals and plants which have become extinct such as Sea Mink, Tasmanian Tiger, West African Black Rhinoceros and so on. In many cases this decaying nature is pre-

vailing but does not come to light. This gradual going off detection is our main purpose of proposing new type of pattern.

1.2 Contributions

- We have proposed a concept of new type of representative patterns named “Decaying patterns” which represents those which were once in frequent pattern set but decayed with time. This could be put into representative pattern set but due to fall off nature, they fail to remain there.
- We have developed an algorithm to mine this type of patterns from real life large datasets which are collected from famous data mining repositories [FIMI](#) and [SPMF](#).
- Data that we have used to test are considered as data stream so we have divided it into set of windows and for any current window we have observed whether it is frequent or erasable which assures getting recent result always.
- We have run our algorithm on six real life datasets and two synthetic datasets. Further our own web service to collect news from prominent online newspapers provided us with floods of daily news. We have pre-processed that huge data and applied our algorithm. From all of these real life large datasets, significant number of decaying patterns have been observed.

The rest of the paper is organized as follows. In Sect. 2, some overview of related works on representative pattern has been given. Section 3 consists of our proposed approach, algorithm and a small simulation to exemplify it. Section 4 contains the experimental analysis based on different performance metrics using the algorithm on many real life and synthetic datasets. Finally, in Sect. 5, we concluded with discussion on the future scope of robustness of our algorithm.

2 Related Work

Maximal Frequent Itemsets [3]: Low *min_sup* generates large number of patterns. Bayardo [3] proposed for storing long patterns (maximal frequent itemsets) in roughly linear scale. If a pattern is X is frequent, all Y where $Y \subset X$ is frequent.

FPclose [4]: Implements another distinction of FP-tree known as CFI-tree (Closed Frequent Itemset Tree). Four fields are necessary for this tree structure - item name, count, node link and level. Subset test of maximality is done with level. Count works for checking if the support count is equal to it’s superset and if it is not, both superset and subset are stored in memory. FP-close is the fastest among the algorithms of that time when minimum support is low but when minimum support is high it becomes slow than Apriori.

TFP [12]: For mining top-k frequent closed items, TFP is an efficient algorithm.

The common factor among all approaches of frequent pattern mining is the usage of *min_sup* threshold which ensures generation of accurate and entire set of frequent item sets which leads to two problems stated below -

First, an appropriate *min_sup* is taken as input but this requires detailed knowledge on mining query. Again setting min support is quite problematic in the sense that a too small threshold may produce thousands of itemsets on the other hand a too big threshold may generate no answers.

Second, Due to the downward closure property, when a long itemset is mined, it may generate an exponential number of itemsets.

To solve these problems they proposed a new approach of mining top-k frequent closed itemsets of minimum length *min_l*, where k is user given number of frequent closed itemset that they want to be mined. k is easy to specify and top-k means k most frequent closed itemsets. *min_l* helps to mine long itemset without mining the short ones first.

ECP (Erasable Closed Pattern) [11]: In factories, for the optimization of production plans erasable pattern (EP) mining plays an important role. For efficient mining of these patterns various algorithms have been proposed. Nevertheless, number of EPs becomes numerous because of large threshold values which cause memory usage overhead. Hence it becomes requisite to mine compressed EPs representation. This paper first came up with the concept of erasable closed patterns (ECPs). These ECPs can be represented without losing information. They at first gave a theory to detect ECPs based on a structure name dPidset and proved it. Then two efficient algorithms, ECPat and dNC-ECPM are proposed. Their result of experiment on these two algorithm shows that for sparse datasets ECPat performs the best but ECPM algorithm is more efficient in the case of memory usage and runtime for rest of the datasets.

DSTree (Data Stream Tree) [8]: In this paper, Leung et. al. proposed the concept of data stream tree. Transactions are sorted in any canonical order chosen by user. Each node keeps a list for frequency count. With the appearance of a new batch of transaction, it is appended to the list to each of the node and frequency count of that node in the current batch. The last entry of a certain node N is the frequency count of that node in the current batch. When the next batch of transactions arrives after fulfilling the batches in a window, the list is shifted to left to place the newest batch to be added as the most recent one. In DSTree costly deletion is not required, only shifting and updating the frequency list will suffice to update tree.

3 Our Proposed Approach

Decaying patterns are important and useful for analysis and many other purposes in many data repository. To the best of our knowledge, it is the first approach for mining decaying pattern. By observing the enormous application in many sectors, we hope these patterns will be effective and useful for gaining knowledge.

As we proposed our algorithm to work on stream of data, we used DSTree based algorithm which serves pretty well for the data stream. Again we have proposed a pattern tree approach to mine decaying patterns and sub patterns of different length, which is also a significant part of our algorithm. We consider

sliding window technique for capturing data effectively. We also thought about including dynamic future of window and batch which we'll adopt as our future contribution.

3.1 Preliminaries

A window is comprised of a set of batches and a batch consists of a set of transactions.

- *win_F* (Frequent window length in decaying pattern) The number of consecutive windows where a pattern has to be frequent.
- *win_{IF}* The number of consecutive windows where a pattern needs to be infrequent. (Difference between total window size & *win_F*).
- *min_{WT}* (Minimum window support threshold) A threshold that is to be crossed by a pattern by occurrence to be frequent in a window.
- *ET* (Error threshold in *win_F*) The maximum number of batches in *win_F* where a pattern can be infrequent.
- *ET'* (Error threshold in *win_{IF}*) The maximum number of batches in *win_{IF}* where pattern can be frequent.

Definition 1 (Decaying pattern). Consider a set of batches $B = \{b_1, b_2, b_3, \dots, b_n\}$ where each batch consists of a number of transactions $T = \{t_1, t_2, t_3, \dots, t_m\}$. If a pattern is frequent (meets *min_{WT}*) in frequent windows (each window of *win_F*) and then becomes infrequent in decay windows (each window of *win_{IF}*), it is called decaying pattern.

e.g. a pattern with a window length 9 and the frequency list of the pattern in these windows is [9, 5, 10, 4, 3, 3, 2, 0, 0]. Let the *min_{WT}* is 4, frequent window length *win_F* is 4. Here the for first four window [9, 5, 10, 4] the pattern meets *min_{WT}*(4). The pattern remains infrequent for last five windows *win_{IF}* [3, 3, 2, 0, 0]. So this pattern is our desired decaying pattern.

– Batch size and window size generation

The size of batch and windows will be based on the number of transaction in a database. In a database with small transactions this value should be kept as small as possible. If user wants to observe the decaying characteristics intensely then he should keep batch size small because the window will move slowly in that case. In general case, as the batch size increases, number of decaying patterns also increases. For sparse dataset, batch size should be as large as possible for this reason.

– *win_F* and *win_{IF}* generation

For dense dataset

$$win_F = \frac{Total\ window\ size}{2}$$

or

$$win_F = \frac{2 \times Total\ window\ size}{3}$$

For sparse dataset

$$win_F = \frac{Total\ window\ size}{3}$$

$$win_IF = Total\ window\ size - win_F$$

But this can be tuned by user according to his demand.

– Minimum window support min_WT

We represent the minimum window support as a percentage value. A minimum window support threshold of 60% means a pattern has to appear in at least 60% of all the transactions in current window of the data stream. The min_WT value is calculated as follows:

$$min_WT = min_sup \% \times number\ of\ transactions\ in\ current\ window$$

– ET and ET' generation

The error threshold in win_F and win_IF should be 1% for dense dataset which refers that a pattern which remains frequent in at least 99% windows of win_F and remains infrequent in at least 99% windows of win_IF , patterns will be accepted as decaying pattern. For sparse dataset ET and ET' can vary from 10% to 20%.

3.2 Tree Construction and Mining

We have used FP-Growth [5] for mining frequent patterns in each window, additionally, as data is considered as stream, mechanism of DSTree [8] is followed for sliding window formation which has been modified according to our purpose.

The transactions in the tree are sorted in frequency descending order for static database. In case of data stream any canonical order can be followed such as alphabetic order or order based on any specific property. With every node in the tree a frequency list is added which contains the frequency of current batches of that item. A batch of transactions is inserted at a time and frequency of each item is appended to the frequency list of each node. When new batch is added, the list is shifted to left and the oldest batch frequency is removed. This has the same effect as deleting the transactions from the oldest batch in a window.

Frequent patterns from each window are added to a new pattern tree. Each node in a pattern tree also consists of a frequency list. Frequent patterns generated from each window of data stream tree are considered as a batch of transactions for the pattern tree. The frequency list of each node contains the frequency of that node in a particular batch. Each batch of frequent patterns are inserted at a time into tree.

When a complete pattern tree is generated, we extract the patterns of our interest by checking only the leaf node. If a leaf node meets the condition of being a decaying pattern the whole branch and the sub-branch of a path including the leaf node will be considered as decaying pattern.

Example Workout

- In Table 1 a small transaction dataset is shown. There are seven items {a, b, c, d, e, f, g} and fifteen transactions. We have divided the transactions into five batches.

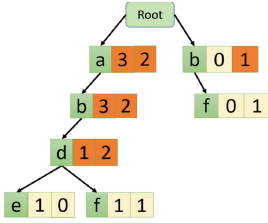
Each batch consists of three transactions, batch length BS is 3. Each window consists of 2 batches, window size WS is 2. We consider our minimum window support threshold min_WT 3 which means a pattern will be frequent in a window if the total frequency of a pattern in the batches of that window is at least 3.

Here, frequent window length in decaying pattern, win_F is 2. So a decaying pattern will be frequent in consecutive 2 frequent windows.

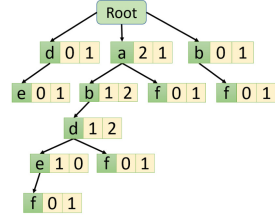
- Now we have to construct a DSTree with these batches. In Fig. 1a the tree is constructed with batch-1 and batch-2. The frequency of each batch is inserted in the frequency list with each node. e.g. ‘a’ has value 3 and 2 in its list which refers to the frequency of ‘a’ in first batch is 3 and in second batch is 2. In each window, we mine frequent pattern from the tree with FP-growth algorithm. The total frequency of ‘a’ in window 1 is 5 which is greater than min_WT , so ‘a’ is a frequent pattern for window 1. In this window, we have found {a}, {b}, {d}, {a, b}, {b, d}, {a, d}, {a, b, d} as frequent patterns.

Table 1. Transactions are arranged in frequency descending order

Batch	Transactions	Contents
First	t ₁	{a, b, d, e}
	t ₂	{a, b, f, d}
	t ₃	{a, b}
Second	t ₄	{a, b, d, f}
	t ₅	{b, f}
	t ₆	{a, b, d}
Third	t ₇	{a, b, d, e, f}
	t ₈	{d, e}
	t ₉	{a, f}
Fourth	t ₁₀	{a, d, f}
	t ₁₁	{d, e, g}
	t ₁₂	{a, e, f, g}
Fifth	t ₁₃	{a, d}
	t ₁₄	{a, d, f}
	t ₁₅	{d, e, g}



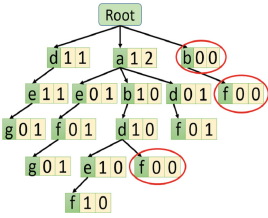
(a) Tree after including first two batch



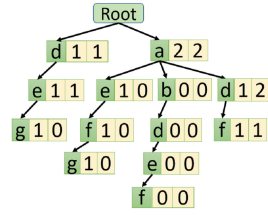
(b) Tree after inserting 3rd batch

Fig. 1. Window 1 & 2

- For second window, we have to insert third batch by removing the oldest batch from the tree. We shifted the frequency list of each node to left and added the frequency of new batch to the right. Again by mining frequent patterns we got {a}, {b}, {d}, {f}, {a, b}, {a, d}, {b, d}, {a, b, d} (Fig. 1b).



(a) Tree after inserting 4th batch



(b) Tree after inserting 5th batch

Fig. 2. Window 3 & 4

- For third window (Fig. 2a), after inserting fourth batch we got {a}, {e} as frequent patterns.
- On fourth window (Fig. 2b) we got {a} and {g} as frequent patterns.

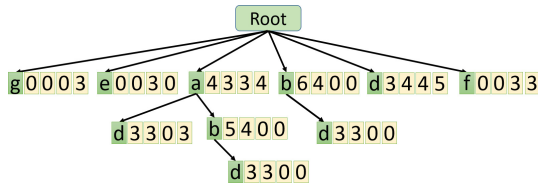


Fig. 3. Pattern tree with the frequent patterns from all windows

- We have built a pattern tree with all the patterns found above, considering the patterns from each window as a batch of transactions to insert in the tree. Each node also consists of a frequency list (Fig. 3).
- Now from this pattern tree, we can easily find out desired pattern only by checking the leaf nodes. The decaying patterns are - $\{a, b, d\}$, $\{b, d\}$, $\{a, d\}$ (Fig. 3).

Algorithm 1: Algorithm for Mining Decaying Patterns

Input : transactions[] where each transaction consists of items, min_sup , win_F and win_IF , ET and ET'

Output: List of decaying patterns in structure named *Decay_Patterns*

```

1 Sort transactions[] in lexicographic order
2  $root \leftarrow \mathbf{Add\_Batch\_to\_Tree}(Batch_0, \dots, Batch_m)$ 
3  $PatternSet \leftarrow \mathbf{FPgrowth}(tree)$ 
4 foreach Remaining batch  $Batch_i$  do
5   |  $\mathbf{Add\_batch\_to\_tree}(Batch_i, root)$ 
6   |  $Pattern\_set[window++] \leftarrow \mathbf{FP\_growth}(tree)$ 
7   |  $\mathbf{Pattern\_tree}(pattern\_set, root)$ 
8 end
9  $Decay\_Patterns \leftarrow \mathbf{Extract\_Pattern}(root)$ 
10 Function  $\mathbf{Extract\_Pattern}(root)$ 
11   | foreach leaf node in tree do
12     | if  $ItemFreq$  in  $win\_F \geq min\_WT$  and  $ItemFreq$  in each batch of  $win\_IF$ 
13     |   |  $== 0$  then
14     |     | keep the node in tree
15     |     |  $Decay\_Patterns[count++] = \beta \cup \text{leaf node}$ 
16     |     | Prune the leaf node upto root;
17     |   | else
18     |     | end
19   | return  $Decay\_Patterns$ 

```

4 Experimental Results

We tested our algorithm on six real life and two synthetic datasets- chess, mushroom, pumsb, accidents, connect, c73d10k, c20d10k. Also we have a web service where we fetch data from some prominent online news portal of Bangladesh. We collected around 59,033 news in 4 months (August to November). We processed the data and applying our algorithm, got our desired result. The algorithm is implemented in JAVA and experiments are performed in Linux environment (Ubuntu 16.04), on a PC with Intel(R) Core-i3-4005U 1.70 GHz processor 4 GB main memory. As there is no literature on finding decaying pattern, we could not compare our result with any other algorithm. For this reason we have shown our result on five metrics.

Algorithm 2: Algorithm for Mining Decaying Patterns

```

19 Function Add_Batch_to_Tree(batch[], root)
20   if root is NULL then
21     foreach batchi in batch[] and transaction t in batchi do
22       |   add t to tree and nodeFrequency to list
23     end
24   else
25     Shift left each node in tree
26     foreach batchi in batch[] and each transaction t in batch0 do
27       |   add t to tree
28       |   add frequency of node to list
29     end
30   end
31   if frequency of each window is 0 of any node in tree then
32     |   delete node and its successors
33   end
34   return root

35 Function Pattern_Tree(patterns[])
36   foreach patterni in patterns[] do
37     |   Add patterni to tree
38     |   insert frequency of node to freq_list
39   end

```

- Number of patterns with varying minimum window support
- Number of patterns with varying window size
- Number of patterns with varying batch size
- Runtime with varying minimum support
- Maximum memory usage with varying minimum support.

4.1 Pattern Count w.r.t. Minimum Window Support

With varying minimum support value, number of decaying patterns also varies. Number of patterns and minimum support change proportionally to each other but in case of decaying pattern this trend does not hold always. Plausibly because when minimum support is low, the patterns tend to be frequent in the decaying window (*win_IF*) and those patterns are rejected as per the definition of decaying pattern. When minimum support is higher those rejected patterns are added in decaying pattern list. From Fig. 4, result of two dense datasets (Fig. 4b and a) connect and mushroom are shown where pattern number decreases with increasing *min_WT* while results of sparse dataset accident and c20d10k (Fig. 5b and a respectively) are different. Number of patterns tends to increase with increasing *min_WT* for accidents. Nearly reverse nature is noticed for c20d10k. So this variation actually depends on dataset.

Our news dataset was highly sparse as we got news of four months only. For better output we splitted the dataset by taking two months' news in a

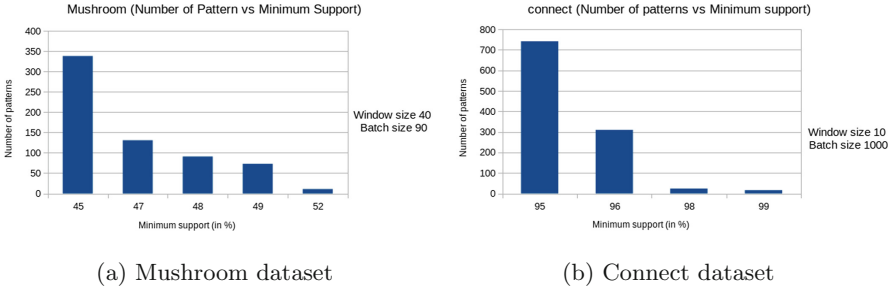


Fig. 4. Pattern number vs Minimum window support (min_WT)

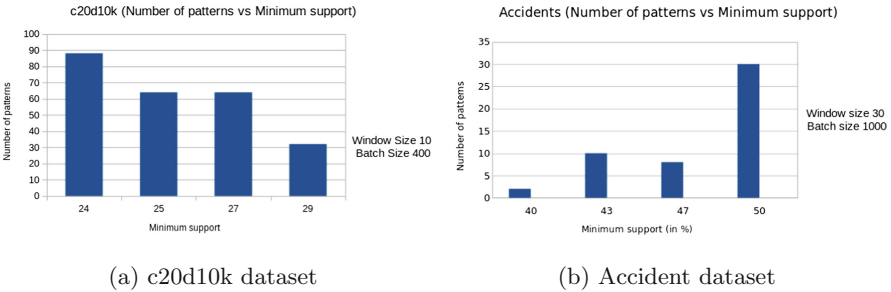


Fig. 5. Pattern number vs Minimum window support (min_WT)

group. From the news of August and September, we observed several decaying patterns (Fig. 6a). Most of them are murder incidents including 16th amendments of Bangladesh constitution, floods in northern part of Bangladesh. From August to November (Fig. 6b), the important decaying news mostly are rape and murder case including some international matters like Rohingya issue, issue of mosque Al-Aqsa in Palestine etc.

4.2 Pattern Count w.r.t. Window Size

In case of dense dataset generally pattern number tends to increase with increasing window size (Fig. 7a) because in larger window, longer decaying pattern can be generated and in that case there will be lot of sub patterns. But in case of sparse dataset the opposite tendency is noticed (Fig. 7b) because sparse dataset contains small number of decaying patterns and with increase of window size possibility of being frequent in a large set of transaction decreases. For this reason, many patterns are rejected as they are infrequent in the frequent windows. This characteristic varies with dataset.

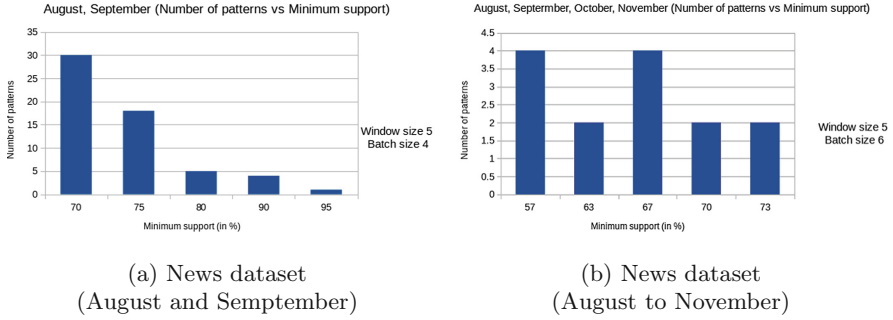


Fig. 6. Pattern number vs Minimum window support (min_WT)

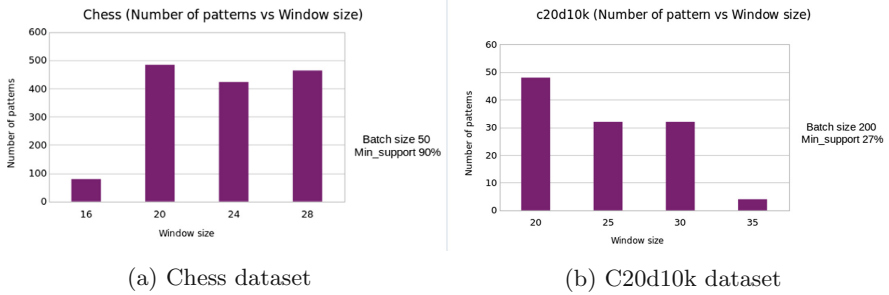


Fig. 7. Pattern number vs Window size

4.3 Pattern Count w.r.t. Batch Size

The next metric is number of patterns w.r.t batch size. Similar characteristic is observed as to previous metric. In case of dense dataset (Fig. 8a) number of patterns increase with increasing batch size because of longer pattern generation and when batch size increases total number of window decreases. So a pattern has to be frequent and infrequent in small number of windows. This increases the probability of getting decaying pattern with increasing batch size. Again, for sparse dataset (Fig. 8b) number of patterns tends to decrease because with batch size is larger window slides faster. In case of sparse dataset, frequent patterns in one window tend to become infrequent in subsequent windows which result in small number of patterns. This characteristic also varies with dataset as depicted in the figures. So we cannot conclude a rigid relationship between pattern count and batch size in mining decaying patterns.

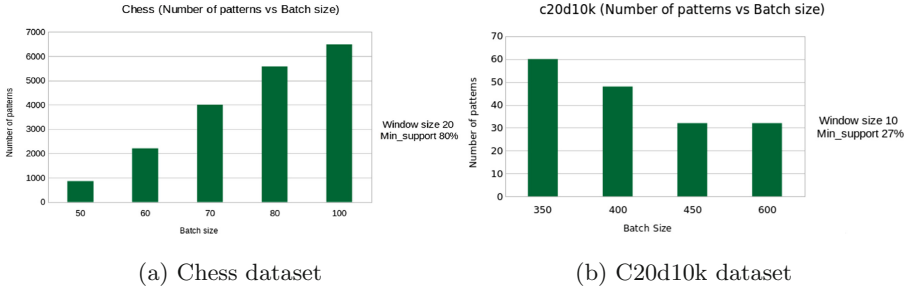


Fig. 8. Pattern number vs Batch size

4.4 Runtime Evaluation

Runtime graphs are shown with varying min_WT on three batches for a dense [Chess] (Fig. 9a) and a sparse dataset [c20d10k] (Fig. 9b). Run time depends on the number of patterns generated and total number of windows to calculate. From the graph, it is comprehensible that run time increases with decreasing minimum support and batch size. For sparse dataset, result is a bit different. With increasing batch size, number of patterns decreases as before which requires greater runtime. In case of sparse dataset, pattern number changes differently with varying win_WT so as the runtime.

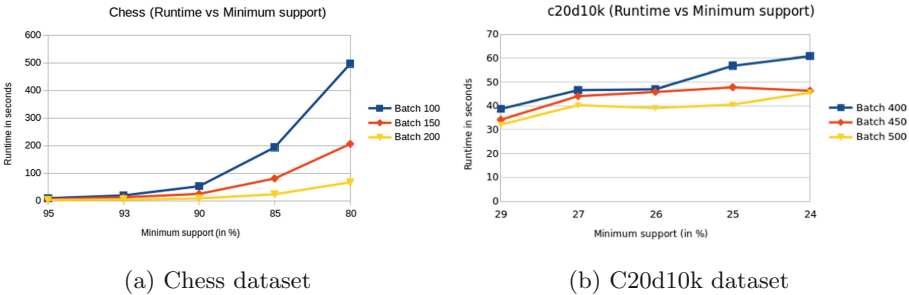


Fig. 9. Runtime (in second) vs win_WT

4.5 Maximum Memory Usage Evaluation

Memory usage also depends on the number of patterns generated. With varying minimum support, the variation of maximum memory usage during run time is shown. We determined maximum memory usage for any instance. During the execution of the code, we have kept the maximum value of memory usage. If at any instance more memory is used than the value, we have updated it. Here we have shown memory consumption of two dataset- chess (Fig. 10a), the dense one and c23d10k (Fig. 10b), the sparse one.

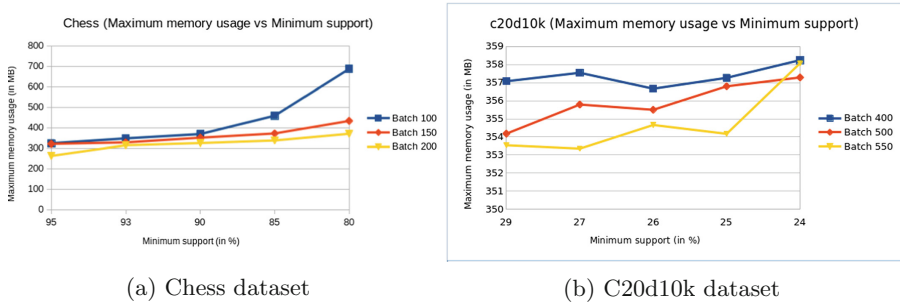


Fig. 10. Maximum memory usage (in MB) vs $win.WT$

5 Conclusions

Many significant patterns are stale with time which should be in representative pattern and need proper attention. This type of patterns are important because if we only focus on those which are always in representative set, some promising patterns that suddenly started decaying will remain neglected. For this, mining this type of patterns are important for different contexts. We have developed an algorithm for mining decaying patterns and applied it by merging the concept of data stream. We have constructed pattern tree structure which speeds up the mining process. As we are dealing with data stream, more interesting knowledge can be found at any instance of time from the patterns.

The application field of this algorithm is huge beginning from market basket data to find new characteristics in dataset. We applied the work on many real life dataset and have got expected results. Our work is highly applicable for mining decaying news and we have depicted significant result from our own processed news data. As a future work, we are planning to develop more compressed structure of tree, applying more efficient mining methodology. We will also carry out more research for generalizing the algorithm so that it can be performed with dynamic window adjusting feature to get the best result without user's input.

References

1. Amphawan, K., Lenca, P., Surarerks, A.: Efficient mining top-k regular-frequent itemset using compressed tidsets. In: Cao, L., Huang, J.Z., Bailey, J., Koh, Y.S., Luo, J. (eds.) PAKDD 2011. LNCS (LNAI), vol. 7104, pp. 124–135. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28320-8_11
2. Amphawan, K., Lenca, P., Surarerks, A.: Mining top-k regular-frequent itemsets using database partitioning and support estimation. *Expert Syst. Appl.* **39**(2), 1924–1936 (2012)
3. Bayardo, R.J.: Efficiently mining long patterns from databases. In: Proceeding of the ACM-SIGMOD International Conference on Management of Data, pp. 85–93 (1998)

4. Grahne, G., Zhu, J.: Fast algorithms for frequent itemset mining using FP-trees. *IEEE Trans. Knowl. Data Eng.* **17**(10), 1347–1362 (2005)
5. Han, J., Pei, J., Yin, J.: Frequent patterns without candidate generation a frequent-pattern tree approach. *Data Min. Knowl. Disc.* **8**(1), 53–87 (2004)
6. Han, J., Wang, J., Lu, Y., Tzvetkov, P.: Mining top-k frequent closed patterns without minimum support. In: *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002)*, Maebashi City, Japan, 9–12 December, pp. 211–218 (2002)
7. Lee, G., Yun, U., Ryang, H.: Mining weighted erasable patterns by using underestimated constraint-based pruning technique. *J. Intell. Fuzzy Syst.* **28**(3), 1145–1157 (2014)
8. Leung, C.K.S., Khan, Q.I.: DSTree: a tree structure for the mining of frequent sets from data streams. In: *Proceedings of the Sixth International Conference on Data Mining (ICDM 2006)*, pp. 928–932. IEEE Computer Society, Washington, DC (2006)
9. Nguyen, G., Le, T., Vo, B., Le, B.: Discovering erasable closed patterns. In: Nguyen, N.T., Trawiński, B., Kosala, R. (eds.) *ACIIDS 2015. LNCS (LNAI)*, vol. 9011, pp. 368–376. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-15702-3_36
10. Nguyen, G., Le, T., Vo, B., Le, B.: EIFDD: an efficient approach for erasable itemset mining of very dense datasets. *Appl. Intell.* **43**(1), 85–94 (2015)
11. Vo, B., Le, T., Nguyen, G., Hong, T.: Efficient algorithms for mining erasable closed patterns from product datasets. In: *IEEE Access*, p. 1 (2017)
12. Wang, J., Han, J., Lu, Y., Tzvetkov, P.: TFP: an efficient algorithm for mining top-k frequent closed itemsets. *IEEE Trans. Knowl. Data Eng.* **17**(5), 652–664 (2005)



An Effective Ensemble Method for Multi-class Classification and Regression for Imbalanced Data

Tahira Alam^(✉), Chowdhury Farhan Ahmed, Sabit Anwar Zahin,
Muhammad Asif Hossain Khan, and Maliha Tashfia Islam

Department of Computer Science and Engineering, University of Dhaka,
Dhaka, Bangladesh
tahiradu@gmail.com, {farhan,asif}@du.ac.bd, sgtlaugh@gmail.com,
maliha.tashfia@gmail.com

Abstract. In the field of Data Mining, classification and regression plays a vital role as they are useful in various real-life domains. Most of the real-life data suffer from data imbalance problem. The performances of the standard algorithms are hindered for the data imbalance problem. A number of methods have been introduced for imbalance data classification. However, most of them are designed for binary class imbalance problems. Furthermore, they suffer from various problems like loss of useful information, likelihood of overfitting, unexpected mistakes etc. On the other hand, data imbalance problem exists in regression analysis also, although very few existing methods consider this problem. Hence, we propose an effective recursive based ensemble method for multi-class imbalance data classification. We also extend our method to propose an effective recursive based method for solving the data imbalance problem in regression. Extensive performance analyses show that our proposed approach achieves high performance in multi-class classification on class imbalance data and regression analysis on skewed or imbalance data. The experimental results also show that our method outperforms various existing methods for imbalance classification and regression.

Keywords: Classification · Multi-class classification · Regression
Imbalance problem

1 Introduction

Classification [11, 16] is a two-step process. In the first step, a classifier is built using a predetermined set of data classes. This is the learning step or training phase, where a classification algorithm builds the classifier by learning from a training set made up of database tuples and their associated class labels. The class label attribute is discrete-valued and unordered. It is categorical where each value serves as a category or class. In the second step, new data or test data, whose class label are unknown, are classified using the classification model [10].

Predicting numeric values is known as regression. Suppose, we want to have a system that predicts the price of a car. Inputs are the car attributes brand, year, engine capacity, mileage etc. that affect a car's price. The output is the price of the car. Surveying the past transactions, we can collect a training data and the machine learning program fits a function to this data to learn new test data [6].

When at least one of the classes is outnumbered by the other classes, then it is called class imbalance data. In multi-class imbalance problem one or some of the classes have much more samples in comparison to the others. Various real life data suffers from multiclass imbalance problem. For example, in a medical diagnosis center the number of Cancer patients is outnumbered by the number of Flu or Hepatitis B patients. Most of the traditional classification algorithms assume that the sample distribution among various classes is balanced. However, data imbalance problem occurs in regression analysis also. Naturally, all real-life data follows the normal distribution. The lower and upper range data are often outnumbered by the middle range data. For example, if we consider the sugar level of diabetes patients, the number of patients with significantly higher or lower sugar level will be lower than the number of patients having normal sugar level. Therefore, if we want to learn from these data then it is vital to solve the imbalance problem in both cases. However, very few existing works address this problem. Therefore, We have proposed an effective ensemble method for addressing multi-class imbalanced problem which converts the multi-class imbalance problem into a number of balanced problems using effective recursive based data partitioning techniques. We have also extended our algorithm in a recursive approach for regression of imbalance data. To the best of our knowledge no such recursive based approach has been proposed for solving the imbalance problem in regression. We have also proposed two new measures for imbalance multi-class classification.

In summary the contributions of this paper are as follows:

- An effective ensemble method for addressing multi-class imbalanced problem has been proposed.
- The data imbalance problem in regression analysis has been addressed.
- We introduce new measures for calculating the recall and precision of multiple minority classes.
- Real-life applications of such approach indicate the suitability of the research work.
- Extensive experimental analyses prove the supremacy of proposed approach.

The remaining part of this paper is organized as follows. In Sect. 2, we present some of the most related works to get insight of our proposed work. The proposed method with the procedures and examples is provided in Sect. 3. We have extensively analyzed our work experimentally to prove its supremacy and Sect. 4 is devoted for this purpose. Finally, in Sect. 5 we make our concluding remarks with some discussions and future plan.

2 Related Works

The approaches to deal with imbalanced data recognition can be summarized into the following two categories: External Methods or Data Level Approaches and Internal Methods or Algorithmic Level Approaches. External methods resize the training datasets in order to balance all the classes. The external methods have the advantage of independence on the underlying classification algorithms. External methods include sampling methods, bagging, SMOTE etc. As our proposed method is also external, we will focus more on the external methods.

Several variations of sampling methods have been proposed for imbalance classification. Japkowicz [11] discussed two strategies: under-sampling and resampling stating that both the sampling approaches were effective. She also observed that for solving the class imbalance problem, using the sophisticated sampling techniques does not provide any advantage. Mani [16] also observed that complicated under-sampling strategies are outperformed by the random under-sampling strategy. A heuristic under-sampling strategy named One-Sided Selection is proposed by Kubat [13] which eliminates the majority class instances that are either borderline or noisy. In addition to the under-sampling strategies, the over-sampling strategies are also used for dealing with the class imbalance problem. SMOTE [4] is a popular method of over-sampling which generates synthetic instances of the minority class. The borderline-SMOTE [9] proposed by Han oversamples the minority class instances near the borderline.

Wang and Yao [19] combined some special sampling methods with Bagging including UnderBagging, OverBagging and SMOTEBagging. Sun [17] proposed novel ensemble methods called SplitBal and ClusterBal for class imbalance data, which are different from Sampling, Bagging or any other external methods, but it only works for binary class data.

Internal or algorithmic level approaches focus on carrying out modification on existing algorithms to strengthen their ability of learning from minority class [7, 14]. Haixiang proposed an ensemble algorithm called BPSO-Adaboost-KNN [8] where the main idea is to integrate feature selection and boosting into ensemble. Chawla [5] proposed a novel approach SMOTEBoost for learning from imbalanced datasets on the basis of the SMOTE algorithm and the boosting procedure. EasyEnsemble [15] is also based on sampling and boosting technologies. EasyEnsemble employ random under-sampling method on majority class. The ensemble algorithm proposed by Krawczyk [12] is based on a cost-sensitive basic classifier and also uses stochastic evolutionary algorithm to fusion basic classifiers. Most of the existing methods suffer from the problems associated with sampling as most of them use sampling for balancing data. Boosting based ensemble methods may alter the original data class distribution as they use sampling methods to increase the minority instances or eliminate the majority class instances.

There are many algorithms for regression analysis [1, 3] but very few address the imbalance problem. Linear Regression is the simplest form of regression. For a response variable y , and a single predictor variable x , it models y as a linear function of x [10]. Another popular method is CART [2] or Classification

and Regression Tree. The representation for the CART model is a binary tree. An algorithm called Piecewise Regression has been proposed for the situation where independent variables, clustered into different groups, exhibit different relationships. But this algorithm does not solve the data imbalance problem.

Another method for regression analysis is Locally Weighted Regression. It is a memory-based method that performs a regression around a point of interest using only training data that are ‘local’ to that point. Torgo [18] used sampling methods for solving the imbalance problem in regression but sampling techniques causes problems like over-fitting.

We have proposed an effective ensemble method for addressing multi-class imbalanced problem which converts the multi-class imbalance problem into a number of balanced problem using effective recursive based data partitioning techniques. We have also extended our algorithm in a recursive approach for regression of imbalance data. To the best of our knowledge no such recursive based approach has been proposed for solving the imbalance problem in regression. We have also proposed two new measures for imbalance multi-class classification.

3 Proposed Method

In this section we have presented our proposed algorithms for multi-class imbalance classification and data imbalance regression. We have described the procedures along with examples to better describe our algorithms.

3.1 Method for Multi-class Imbalance Classification

Our method for multi-class imbalance classification converts the multi-class imbalance problem into a number of balanced problems. The process is repeated recursively, bounded by a specified threshold. Our proposed method is different from other data balancing methods like sampling, Underbagging etc. because of different data balancing techniques. Moreover, our unique recursive based approach effectively partitions the imbalanced data into balanced data. Our algorithm is described below:

1. Data Partitioning

In multi-class imbalance data, there can be more than one majority classes and minority classes. But for better understanding we will refer the class with the most number of records(n_{maj}) as majority class, C_{maj} and class with the minimum number of records(n_{min}) as minority class C_{min} . In this step we will partition all the classes except C_{min} , so that each partition contains n_{min} number of records. Here, the total number of partitions for a class C_i containing n_{ci} records will be n_{ci}/n_{min} if $n_{ci} \% n_{min} = 0$, else it will be $n_{ci}/n_{min} + 1$. For data partitioning we have used two methods. They are described below:

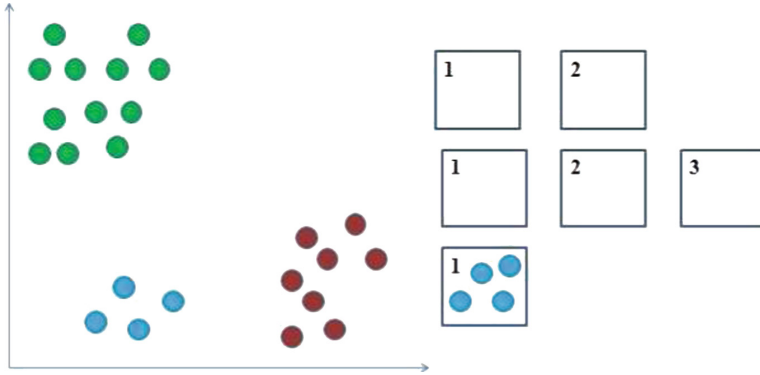


Fig. 1. Multi-class imbalance dataset. (Color figure online)

– **Partition using Balanced Distribution**

For partitioning using balanced distribution the data of a class are partitioned in such a way so that the distribution of the data in the partitions remain the same as the distribution in the class. A record is randomly selected and its $n_{ci}/n_{min} - 1$ nearest record are selected. These $1 + (n_{ci}/n_{min} - 1) = n_{ci}/n_{min}$ records are assigned to the n_{ci}/n_{min} partitions (one record per each partition). This process is repeated n_{ci} times. This ensures that each of the partitions holds all types of data of the class. In other words, each partition represents the class other than representing a part of the class only. After that, if any more records are left then they are assigned to a different partition.

– **Random Partitioning**

In case of random partitioning data are randomly partitioned into n_{ci}/n_{min} or $n_{ci}/n_{min} + 1$ partitions.

2. Create Balanced Data

The data partitions for each class are used for creating a number of balanced data bins. The number of data bins is equal to the number of partitions of the majority class. The partitions of the majority class are assigned to the bins, one partition per each bin. The data partitions of the other classes are sequentially assigned to the balanced data bins and repeated if necessary.

3. Building Ensemble Classifier

Each of the balanced data bins are used for building a classifier. All these classification models are used for building an ensemble classifier.

4. Classifying Test Data Using Ensemble Rule

In this step for a new test data, a class label is assigned using the ensemble model. We have used Majority Voting as our ensemble rule.

5. Recursive Approach

In the first step as the rest of the $n_{ci} \% n_{min}$ records are assigned to a new partition, so there may still exist class imbalance problem in these data bins. If so, then the bins are again partitioned into balanced bins by recursive call. Our method is different from other methods for effective data partitioning and unique recursive approach.

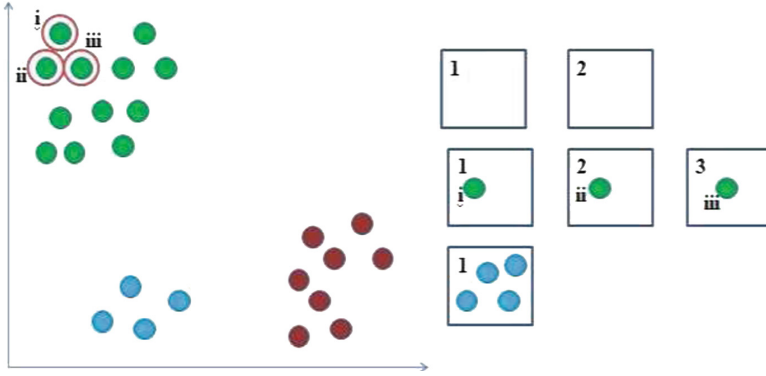


Fig. 2. Partitioning using balanced distribution. (Color figure online)

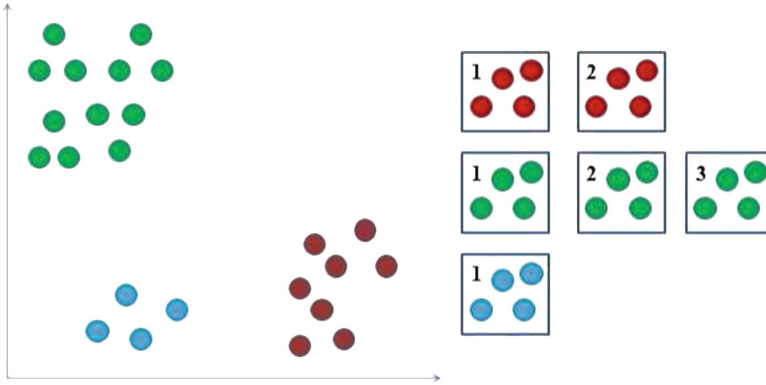


Fig. 3. Partitions of the classes. (Color figure online)

Let us see our proposed method with an example. In Fig. 1, we have three classes indicated by three different colors. Class imbalance problem exists in the data, where the Blue class is the minority class. Hence we do not need to partition the Blue class. But, we need to partition the Red and the Green classes. For the Green class, number of partition, $P = 12 \div 4 = 3$, as $12\%4 = 0$. For Partition using Balanced Distribution, each time we will select $(3 - 1) = 2$ nearest neighbours for a randomly selected data instance. On the other hand for the Red class, number of partition, $P = 8 \div 4 = 2$, as $8\%4 = 0$. Each time we will select $(2 - 1) = 1$ nearest neighbors for a randomly selected data instance. Suppose, for the Green class, we randomly select the data instance indicated with i in Fig. 2. Then we select its two nearest neighbors (indicated by ii and iii). These three data instances are assigned to the three different data partitions, one at each. This will continue until all the three partition size is equal to the size of the minority class, that is the Blue class. Following the same procedures, the Red class is divided into two partitions. In Fig. 3, we can see the partitions of all the classes.

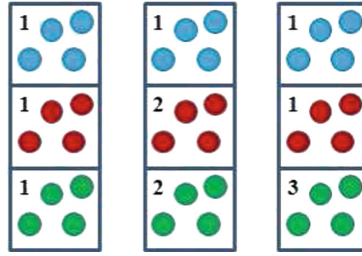


Fig. 4. Balanced data bins. (Color figure online)

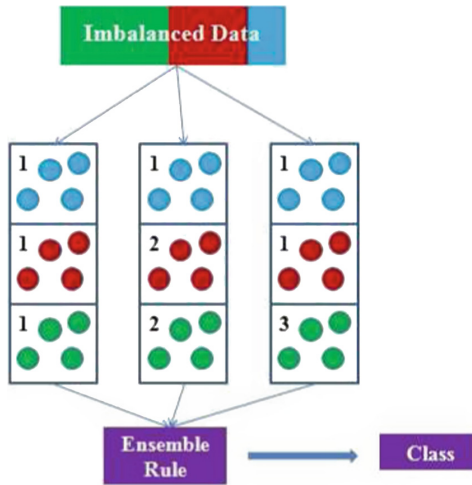


Fig. 5. Building ensemble classifier and classifying new data using ensemble rules. (Color figure online)

In the second step we create a number of balanced data bins using the partitions. The number of balanced data bins is equal to the number of partitions of the majority class. We take the partitions of the majority classes and assign them to the balanced data bins, one at each time. Then we assign the partitions of the other classes sequentially and repeat if necessary. In Fig. 4, we assign the partitions of the Green class to the balanced data bins, one at each. Then we assign the partitions of the Red class (partition 1 and 2) and then repeat partition 1 again. Similarly we assign the only partition of the Blue class to all the bins. Finally we get three balanced data bins. If the data imbalance problem still existed in any of the data bins, then we would have again recursively called the data partitioning and balancing method for the imbalanced bin. However, here all the data bins are balanced. In the third step these balanced data bins are used for creating an ensemble classifier. This ensemble classifier is used for classifying new data using ensemble rule as shown in Fig. 5.

3.2 Method for Solving Data Imbalance Problem for Regression

We have introduced a recursive approach for solving the data imbalance problem of regression. Our proposed recursive approach for solving data imbalance problem for regression is described below:

1. Identifying Data Imbalance

The data imbalance problem is identified by repeatedly dividing the range of values into smaller ranges of equal size and calculating the records of each smaller range. At first, the range is divided into three equal size ranges. If the ratio of the number of records of two smaller ranges is greater than or equal to 1.5, then it is identified that data imbalance problem exists in the data. In that case the data belonging to each range is assigned a group. For further steps, these groups are treated as class labels of the data. If the ration is smaller than 1.5 than the number of smaller range is increased by one and again data imbalanced is checked. The threshold for imbalance ratio is set to 1.5 as the popular data respiratories like Keel classifies a dataset as imbalance if the imbalance ratio among the classes is more than or equal 1.5. It is repeated till a user specified threshold.

2. Building Ensemble Classifier

This step is the same as steps 1–4 of multi-class imbalance data classification. The groups of the data are treated as their class. At the end of this step a group/class label is found. The steps are briefly described below:

– Data Partitioning

Let us suppose, the group with the most number of records($Group_{maj}$) is the majority group, G_{maj} and group with the minimum number of records($Group_{min}$) as minority class G_{min} . In this step we will partition all the groups except G_{min} , so that each partition contains $Group_{min}$ number of records. Here, the total number of partitions for a group G_i containing $Group_{ci}$ records will be $Group_{ci}/Group_{min}$ if $Group_{ci} \% Group_{min} = 0$, else it will be $Group_{ci}/Group_{min} + 1$.

The data of a group are partitioned in such a way so that the distribution of the data in the partitions remains the same as their distribution in the group. A record is randomly selected and its $Group_{ci}/Group_{min} - 1$ nearest record are selected. These $1 + (Group_{ci}/Group_{min} - 1) = Group_{ci}/Group_{min}$ records are assigned to the $Group_{ci}/Group_{min}$ partitions(one record per each partition). This process is repeated $Group_{ci}$ times. This ensures that each of the partitions holds all types of data of the class. In other words, each partition represents the class other than representing a part of the class only. After that, if any more records are left then they are assigned to a different partition.

– Create Balanced Data

The data partitions for each group are used for creating a number of balanced data bins. The number of data bins is equal to the number of partitions of the group having the most number of tuples. The partitions

of that group are assigned to the bins, one partition per each bin. The data partitions of the other groups are sequentially assigned to the balanced data bins and repeated if necessary.

– **Building Ensemble Classifier**

Each of the balanced data bins are used for building a classifier. The group labels of the data are used as their class label. All these classification models are used for building an ensemble classifier.

– **Classifying Test Data Using Ensemble Rule**

In this step for a new test data, a class or group label is assigned using the ensemble model. Here also we have used Majority Voting as the ensemble rule.

3. Predicting the Value

In this step a prediction value is found by applying a algorithm for regression analysis on the data of the selected group/class from the previous step.

4. Recursive Technique

As the predicted value is continuous in case of regression analysis, the data in the selected group can again suffer from data imbalance problem. This problem is solved by repeating the same process on the data of the selected group. This recursive process is repeated for a specified threshold. This recursive based approach makes our method unique and remarkably increases the effectiveness of the algorithm.

Let us see our proposed method with an example. Figure 6 shows a regression dataset where the predicted values are people’s ages. The range of the predicted value is 1 to 60. So we divide the range in three equal smaller ranges and count the number of data instances for each range. The ratio of the smallest and largest number of data is $(9 \div 3) = 3$, which is greater than 1.5. So, it is identified that data imbalance problem exists in the data. Now, each range of data instances is given a group or class label as shown in Fig. 7.

The next step is like the four steps of our method for multi-class imbalance classification. The groups or classes are partitioned so that the size of each parti-

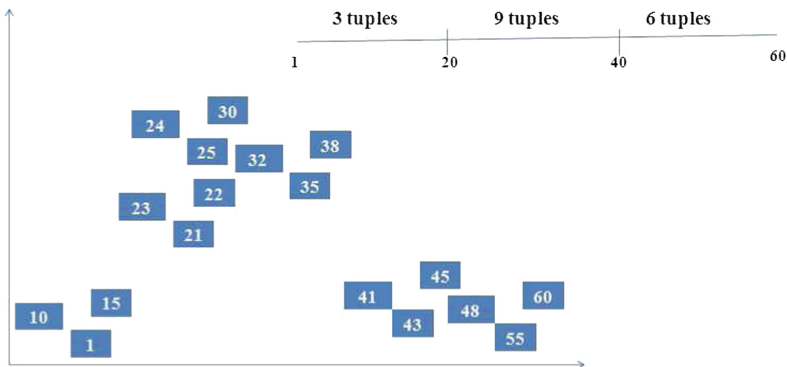


Fig. 6. Identifying data imbalance problem in regression problem.

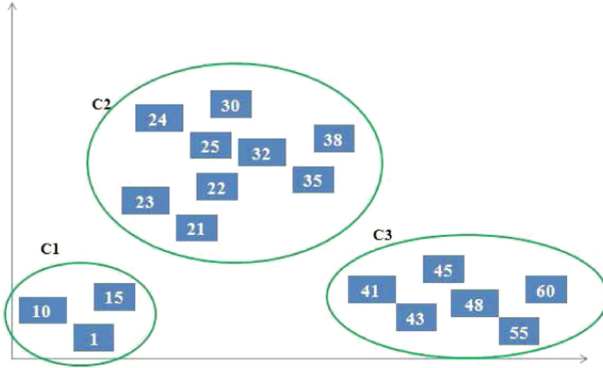


Fig. 7. Partitioning the data for regression into groups.

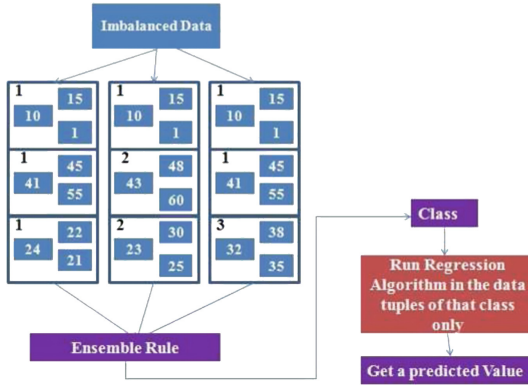


Fig. 8. Balanced regression.

tion is equal to the size of the minority class or group. These partitions are used for creating a number of balanced data bins. As shown in Fig. 8, these balanced data bins are used for building an ensemble classifier. A new data instance is assigned a class or group label. But since this is a regression problem, we need to predict a numeric value. So a standard regression algorithm is applied, but only on the data instances of the group or class that is selected by the ensemble classifier. Suppose, for a new test data, if the ensemble classifier selects the class $C3$, then a standard regression algorithm will be applied on the data instances of $C3$ only. As the predicted values are continuous values, so data imbalance problem may still occur in the selected group. For solving this problem, we repeat the process of selecting the group recursively till a user specified threshold. In that case $C3$ will be further divided into a number of smaller ranges and if there is data imbalance problem, then the whole process will be repeated again. In our experiments we have set the threshold to two.

3.3 Min Average Recall and Min Average Precision

We have also proposed two performance measures for multi-class classification problem. Min Average Recall is the average of the recalls of the minority classes. A threshold (1.5) is set for imbalance ratio for identifying minority classes. Min Average Precision is the average of the precisions of the minority classes calculated in the same way. These measures represents the recalls and precisions of all the minority classes.

4 Experimental Results

In this section we show the experimental results achieved by our proposed algorithms.

4.1 Performance Analysis for Multi-class Imbalance Classification

For evaluating the performance of our algorithm for multi-class imbalance data, we have performed experiments on 7 real life datasets, found the datasets from Keel Dataset Respiratory. They are all multi-class data with class imbalance problem. The datasets are: Balance (Total number of classes: 3, Imbalance Ratio: 5.88), Dermatology (Total number of classes: 6, Imbalance Ratio: 5.55), New Thyroid (Total number of classes: 3, Imbalance Ratio: 4.84), Wine (Total number of classes: 3, Imbalance Ratio: 1.5), Splice (Total number of classes: 3, Imbalance Ratio: 2.16), Led7digit (Total number of classes: 10, Imbalance Ratio: 1.54), Zoo (Total number of classes: 7, Imbalance Ratio: 10.25). Here, the imbalance ratio refers to the ratio of the majority and minority class size.

We have compared our algorithm for multi-class imbalance classification with two popular external methods: SMOTEBagging and SplitBal. SMOTEBagging is a combination of SMOTE and Bagging. It is designed for multiclass imbalance classification problem. On the other hand, SplitBal is designed for binary class imbalance problem. So, we used All Versus All (AVA) method for making it suitable for multi-class problems. The performance measures used for evaluating their performance are Average Recall (average of the Recalls of all the classes), Min Average Recall, Average Precision (average of the Precisions of all the classes), Min Average Precision and F measure. Min Average Recall and Min Average Precision are our proposed performance measure which described in Sect. 3. We have used cross-validation for all the experiments. j48 is used as the base classifier and for ensemble rule, we have used majority voting. The recursion limit is set to 2 and the threshold for checking imbalance is set to 1.5.

Table 1 shows the comparison results for Average Recall on the seven datasets. We can see that our algorithms performs better than the other two algorithms as our algorithm effectively partitions the majority classes so that there is no loss of data or overfitting.

Table 2 shows the comparison results for Min Average Recall on the seven datasets. We can see that our algorithms performs better than the other two algorithms as our algorithm uses effective recursive based data balancing method.

Table 1. Average recall comparison (Base classifier: J48)

Dataset	PBD	RP	SMOTEBagging	SplitBal(AVA)
Balance	74.89	75.17	61.56	69.8
Dermatology	96.89	97.17	96.78	95.03
New thyroid	95.48	95.19	92.67	93.09
Wine	97.59	98.15	93.33	95.47
Splice	95.20	95.24	92.13	92.12
Led7digit	71.76	71.52	70.59	66.23
Zoo	89.05	87.99	85.89	88.19

Table 3 shows the comparison results for Average Precision on the seven datasets. We can see that our algorithms performs better than the other two algorithms as our algorithm uses effective recursive based data balancing method.

Table 4 shows the comparison results for Min Average Precision on the seven datasets. We can see that our algorithms performs better than the other two algorithms as our algorithm effectively partitions the majority classes so that there is no loss of data or overfitting.

Table 2. Min average recall comparison (Base classifier: J48)

Dataset	PBD	RP	SMOTEBagging	SplitBal(AVA)
Balance	91	80.67	14	68.9
Dermatology	96.33	96.73	96.27	94.44
New thyroid	94.67	94.44	92	92.53
Wine	100	100	94.17	97.58
Splice	95.03	95.12	93	91.97
Led7digit	79.64	78.06	76.03	73.71
Zoo	86.94	86.63	86.66	82.78

Table 3. Average precision comparison (Base classifier: J48)

Dataset	PBD	RP	SMOTEBagging	SplitBal(AVA)
Balance	82.69	79.98	55.63	73.85
Dermatology	97.40	97.56	97.09	95.39
New thyroid	95.91	95.72	93.53	93.04
Wine	98.08	98.13	94.03	95.94
Splice	95.25	95.24	93.16	92.25
Led7digit	73.30	73.15	71.86	67.51
Zoo	91.34	92.74	88.45	91.27

Table 4. Min average precision comparison (Base classifier: J48)

Dataset	PBD	RP	SMOTEBagging	SplitBal(AVA)
Balance	58.31	60.05	33.46	56.09
Dermatology	96.79	97.08	96.64	94.92
New thyroid	98.68	98.44	97.11	96.21
Wine	97.68	97.68	98.43	97.59
Splice	95.21	95.26	90.25	90.71
Led7digit	78.18	79.16	74.23	68.90
Zoo	90.12	89.73	87.64	90.04

Table 5. F measure comparison (Base classifier: J48)

Dataset	PBD	RP	SMOTEBagging	SplitBal(AVA)
Balance	78.60	77.50	58.44	71.77
Dermatology	97.14	97.36	96.93	95.21
New thyroid	95.69	95.45	93.10	93.06
Wine	97.93	98.02	93.68	95.70
Splice	95.22	95.24	92.64	92.18
Led7digit	72.52	72.33	71.22	66.86
Zoo	90.18	90.30	87.15	89.70

Table 6. Mean square error for regression analysis

Dataset	Linear reg.	REPTree	LWR	Piecewise reg.	Balanced reg.
compactiv	0.30118	0.03388	0.07098	0.1368	0.02042
concrete	0.11539	0.08307	0.16525	0.09991	0.06705
delta-ail	0.13082	0.14135	0.16409	0.12699	0.08484
delta-elv	0.12855	0.13193	0.16865	0.12704	0.09753
diabetes	0.19487	0.21942	0.19293	0.18675	0.18019
ele-1	0.06014	0.06714	0.08328	0.06824	0.05496
elevators	0.09513	0.09264	0.19109	0.0771	0.06897
friedman	0.10596	0.10597	0.21149	0.06444	0.08095
laser	0.09539	0.05235	0.12432	0.0698	0.041
plastic	0.4283	0.17345	0.31458	0.18267	0.14362
tic	0.64137	0.6429	0.65574	0.63984	0.53042

Table 7. Mean absolute error for regression analysis

Dataset	Linear reg.	REPTree	LWR	Piecewise reg.	Balanced reg.
compactiv	0.21557	0.01328	0.0344	0.0943	0.01043
concrete	0.09239	0.06342	0.1308	0.07975	0.04634
delta-ail	0.09035	0.09486	0.11322	0.08734	0.05661
delta-elv	0.09472	0.09558	0.12043	0.09368	0.06501
diabetes	0.16342	0.18281	0.16108	0.15747	0.14822
ele-1	0.04105	0.04564	0.06458	0.04986	0.03629
elevators	0.0647	0.06763	0.14434	0.04849	0.04934
friedman	0.08642	0.08828	0.19836	0.05223	0.06089
laser	0.06157	0.02838	0.09381	0.02586	0.01933
plastic	0.40027	0.13906	0.28402	0.1316	0.07763
tic	0.47868	0.47589	0.48749	0.47613	0.30613

Table 5 shows the comparison results for F measure on the seven datasets. We can see that our algorithms performs better than the other two algorithms as our algorithm do not suffer from loss of data or overfitting.

4.2 Performance Analysis for Data Imbalance Regression

The performance measures used for evaluating the performance are Mean Absolute Error and Mean Square Error. For all the experiments, the recursion limit is set to 2 and the threshold for checking imbalance is set to 1.5. We have compared our results with four algorithms (Linear Regression, REPTree, Piecewise Regression, Locally Weighted Regression) to indicate the effectiveness of our algorithm.

Tables 6 and 7 shows the results of our algorithm with respect to Mean Square Error and Mean Absolute Error respectively. For the final prediction algorithm, we have used REPTree. We can see that our algorithm performed better than the other four algorithms as our algorithm handles the data imbalance problem by balancing the data.

5 Conclusions

Most algorithms proposed for solving the class imbalance problem are well-suited for binary classification. Moreover they suffer from various problems, like sampling methods alter the original data class distribution of imbalanced data and then lead to some unexpected mistakes. For example, over-sampling might lead to overfitting and under-sampling may drop some potentially useful information. Bagging and Boosting algorithms, for each iteration it may still suffer from the class imbalance problem as the sampled subset in a given iteration has the

similar class distribution with the original data set. Again, the data imbalance problem has not been addressed in most of the algorithms for regression analysis. We propose an recursive based ensemble method for classifying multi-class imbalance data and also a recursive based method for regression analysis for imbalanced data.

We wish to further improve our method. We wish to introduce an efficient ensemble rule suitable for multi-class classification and regression. Our proposed method is external, that is we do not alter the base algorithms. In future we wish to propose an internal algorithm as well which can handle multi-class imbalance problem and regression problem with imbalance data.

References

1. Branco, P., Torgo, L., Ribeiro, R.P.: A survey of predictive modelling under imbalanced distributions. CoRR abs/1505.01658 (2015)
2. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and regression trees. Technical report, Wadsworth International, Monterey, CA (1984)
3. Buza, K., Nanopoulos, A., Nagy, I.J.: Nearest neighbor regression in the presence of bad hubs. *Knowl.-Based Syst.* **86**, 250–260 (2015)
4. Chawla, N.V., Bowyer, K.W., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **16**, 321–357 (2002)
5. Chawla, N.V., Lazarevic, A., Hall, L.O., Bowyer, K.W.: SMOTEBoost: improving prediction of the minority class in boosting. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) PKDD 2003. LNCS (LNAI), vol. 2838, pp. 107–119. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39804-2_12
6. Donmez, P.: Introduction to Machine Learning, 2nd edn. By Ethem Alpaydm. The MIT Press, Cambridge (2010). ISBN 978-0-262-01243-0. \$54/£ 39.95 + 584 p. (*Nat. Lang. Eng.* **19**(2), 285–288 (2013))
7. Galar, M., Fernández, A., Tartas, E.B., Sola, H.B., Herrera, F.: A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Trans. Syst. Man Cyber. Part C* **42**(4), 463–484 (2012)
8. Guo, H., Li, Y., Li, Y., Liu, X., Li, J.: BPSO-Adaboost-KNN ensemble learning algorithm for multi-class imbalanced data classification. *Eng. Appl. AI* **49**, 176–193 (2016)
9. Han, H., Wang, W.-Y., Mao, B.-H.: Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In: Huang, D.-S., Zhang, X.-P., Huang, G.-B. (eds.) ICIC 2005. LNCS, vol. 3644, pp. 878–887. Springer, Heidelberg (2005). https://doi.org/10.1007/11538059_91
10. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann, Burlington (2000)
11. Japkowicz, N.: The class imbalance problem: significance and strategies. In: Proceedings of the 2000 International Conference on Artificial Intelligence (IC-AI 2000), vol. 1, pp. 111–117 (2000)
12. Krawczyk, B., Wozniak, M., Herrera, F.: Weighted one-class classification for different types of minority class examples in imbalanced data. In: 2014 IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2014, Orlando, FL, USA, 9–12 December 2014, pp. 337–344 (2014)

13. Kubat, M., Matwin, S.: Addressing the curse of imbalanced training sets: one-sided selection. In: Proceedings of the 14th International Conference on Machine Learning, pp. 179–186. Morgan Kaufmann (1997)
14. López, V., Fernández, A., del Jesús, M.J., Herrera, F.: A hierarchical genetic fuzzy system based on genetic programming for addressing classification with highly imbalanced and borderline data-sets. *Knowl.-Based Syst.* **38**, 85–104 (2013)
15. Liu, X., Wu, J., Zhou, Z.: Exploratory undersampling for class-imbalance learning. *IEEE Trans. Syst. Man Cybern. Part B* **39**(2), 539–550 (2009)
16. Mani, I.: KNN approach to unbalanced data distributions: a case study involving information extraction. In: Proceeding of International Conference on Machine Learning (ICML 2003) (2003)
17. Sun, Z., Song, Q., Zhu, X., Sun, H., Xu, B., Zhou, Y.: A novel ensemble method for classifying imbalanced data. *Pattern Recogn.* **48**(5), 1623–1637 (2015)
18. Torgo, L., Branco, P., Ribeiro, R.P., Pfahringer, B.: Resampling strategies for regression. *Expert Syst.* **32**(3), 465–476 (2015)
19. Wang, S., Yao, X.: Diversity analysis on imbalanced data sets by using ensemble models. In: CIDM, pp. 324–331. IEEE (2009)



Automating the Extraction of Essential Genes from Literature

Ruben Rodrigues¹, Hugo Costa², and Miguel Rocha¹(✉)

¹ CEB - Centre Biological Engineering, University of Minho, Braga, Portugal
mrocha@di.uminho.pt

² Silicolife Lda, Braga, Portugal

Abstract. The construction of repositories with curated information about gene essentiality for organisms of interest in Biotechnology is a very relevant task, mainly in the design of cell factories for the enhanced production of added-value products. However, it requires retrieval and extraction of relevant information from literature, leading to high costs regarding manual curation. Text mining tools implementing methods addressing tasks as information retrieval, named entity recognition and event extraction have been developed to automate and reduce the time required to obtain relevant information from literature in many biomedical fields. However, current tools are not designed or optimized for the purpose of identifying mentions to essential genes in scientific texts.

In this work, we propose a pipeline to automatically extract mentions to genes and to classify them accordingly to their essentiality for a specific organism. This pipeline implements a machine learning approach that is trained using a manually curated set of documents related with gene essentiality in yeast. This corpus is provided as a resource for the community, as a benchmark for the development of new methods. Our pipeline was evaluated performing resampling and cross validation over this curated dataset, presenting an accuracy of over 80%, and an f1-score over 75%.

1 Introduction

In recent years, organisms modified genetically have been used as hosts in the production of compounds of interest (e.g. biofuels or drugs) through Biotechnology [1,2]. In many cases, these hosts are subjected to specific genetic modifications to design strains that are able to improve productivity or yield in these bio-processes. The identification of essential genes for these microbial hosts is an important task within this effort.

In this context, several repositories with manually curated information of organism oriented gene essentiality (e.g. OGEE [3], SGD [4]) emerged to identify which genes can be removed maintaining the modified organism viability, given some experimental conditions (e.g. media). The construction of such repositories requires a large amount of information that is spread in different sources, mainly scientific literature [3,4]. The extraction of relevant information from literature

about essential genes is a time consuming task, requiring a huge amount of manual curation from researchers.

The Text Mining (TM) field has emerged from the efforts to automate and reduce the required time to retrieve and extract relevant information from literature. This field combines computational approaches applied for several linguistic challenges, as the identification of relevant documents for a specific theme, the recognition of name entities with biological meaning from texts, or the extraction of semantic information of these named entities or relationships/ events relating them [5].

The identification of gene mentions in literature, with a correct association to a specific organism, is a difficult task even when we have a selected set of relevant documents. Named entity recognition (NER) is a TM field that has been used for the identification of biological entities assigned to a class of interest (e.g. proteins, compounds, genes or cell lines) [5–8].

Dictionaries and ontologies have been used as resources for NER, being used as data structures supporting search and matching algorithms (e.g. binary search algorithm), used to annotate free text with specific entries from biological databases and other repositories [6]. This approach has some limitations, being one of them the need to have complete databases for a specific biological context that usually are not enough to match all possible entities (e.g. lack of all possible gene synonyms on a gene dictionary).

NER approaches based on expression rules have been used to match named entity variants, which are not present on dictionaries and ontologies [7]. However, this requires the definition of complex rules created from name patterns for a specific biological context.

Supervised Machine Learning (ML) models, such as Support Vector Machines (SVMs) or Conditional Random Fields (CRFs) have been applied for NER with fast and reliable results [8]. These methods work by training ML models from a large set of data, typically manually curated, which may then be applied to predict the classes of new examples. ML methods have also been used for many other applications in TM.

In the case of NER for genes, we will opt here for the use of dictionaries, since in this particular case they have shown good performance. Indeed, dictionaries for gene names and synonyms for well studied organisms, such as yeast or humans, can achieve good performance in NER tasks. Also, applying a curated dictionary with gene names for a determined organism improves the specificity of the dictionary based NER systems.

When trying to assess gene essentiality from text, a first step of NER to find gene mentions is needed. Then, a second level of information extraction is required to identify if the identified genes are essential or not, given the semantics of the sentence where the gene is mentioned. This will be handled as a problem of Event extraction (EE), another major TM task, which tries to identify semantic interactions between previously annotated entities [3, 5, 9].

We will consider here that the presence of certain keywords, like “essential”, “nonessential” or “viability”, can be important to define the sentence’s

semantics. We will call these keywords as triggers and identify those in sentences also using NER, matching the tokens in the sentence with a set of pre-defined keywords.

Pairing mentions to triggers and genes can give us information on the semantics of the sentence regarding the gene’s essentiality. Here, for each pair, we will provide a classification task seeking to decide if the pair is an essentiality event or not. An event can be defined as a relation between a set of entities with biological interest. In our case, this relationship will be established between a gene and a trigger, and will represent an event of gene essentiality.

As an example, in the sentence “*Disruption of gene A compromises the viability of organism D*”, the entity “*gene A*” and the trigger “*viability*” are associated as a pair, which is classified as an essentiality event “*gene A - viability*” due to the meaning of the whole sentence (the verb “*compromises*” implies that the “*gene A*” is essential for “*organism D*”).

Shallow parsing and dependency parsing methods have been applied to different EE tasks, identifying semantic patterns on sentences that denote the interaction of annotated entities in a sentence [10,11]. Those methods require a set of rules, defined as grammars, that try to represent the structure of a sentence computationally. Those grammars are difficult to obtain because they require large amounts of written text for a specific theme, being normally trained and curated from large text repositories as journal news.

Co-occurrence based methods are able to identify events from associations of entities present in each sentence [12]. Those associations are then classified as event or not regarding a arbitrary rule, a statistical or an ML model. Similarly to ML models applied for NER, ML models applied for EE based on co-occurrences require a large amount of documents manually annotated with events to allow for ML model training.

Training ML models on an EE co-occurrence based system using literature with curated gene essentiality evidences would enable the prediction of novel gene essentiality evidences in unannotated literature. However, previously proposed NER and EE systems lack specificity to extract genes with an essentiality context, since they are designed to identify biological entities and events within a more generic context. Also, there is a lack of adequate manually curated datasets to enable training ML models for this task.

In this work, we aim to help researchers to identify essential gene evidences from literature in an automated way, by proposing a gene essentiality extraction method, which is implemented by a TM pipeline including NER and EE sub-systems. Our method firstly identifies gene names for a specific organism and triggers related to gene essentiality, and then provides ML-based models for the classification of the essentiality gene context present in the sentences where gene-trigger pairs are identified. We have also created a large manually curated corpus for yeast genes essentiality with over 6000 sentences, which allows to train our ML models and provide a benchmark for their validation, as well as a resource for the research community.

In this article, we provide a detailed description of the gene essentiality extraction method proposed in this work. Afterwards, the system performance is evaluated in terms of precision, recall and f1 score using our novel curated dataset. Finally, we discuss the advantages and further improvements of the proposed system.

2 Algorithms and Implementation

2.1 Proposed Algorithm

Our system contains two main steps to extract essential genes: (i) identification of triggers and gene names with a dictionary-based NER approach, and (ii) classification of essential genes with an EE based ML approach. Figure 1 shows the overall pipeline with the two main steps of our system.

The NER for gene identification is performed using a dictionary-based approach that requires a dictionary built with a set of gene names for a specific organism (e.g. *S. cerevisiae* or *E. coli*). The system matches and annotates all dictionary names against the free text using a binary search algorithm.

On the other hand, the triggers are recognized following a similar approach, using a dictionary built from a curated set of names like “essentiality”, “essential”, “nonessential”, “non-essential”, “unessential gene” or other variation keywords that define the essentiality context of gene names in the sentence.

In the second step of the pipeline, an EE sub-system is used to annotate events between the previously identified genes and triggers. Those events are classified as essential, not essential or not related taking into account the sentence meaning. For EE, we use an algorithm designed for general-purpose tasks, which takes pairs of annotated entities and classifies those accordingly to a set of features.

The EE sub-system is separated in two main pipelines: training and prediction. In both pipelines, there is a common step, the feature matrix generation which requires the MALLET framework [13]. This step processes the corpus and converts it into a matrix of features characterizing pairs of entities (which can be labelled as possible events).

The set of features used in our method is provided and briefly described in Table 1. These features can be divided, regarding the information extracted from texts, into two main groups: sentence morphology and event morphology.

Sentence morphology features are extracted from the sentences’ structure, including any information not directly related with the annotated pairs (e.g. sentence length, verbs present in the sentence, etc). On the other hand, event morphology features are related directly with the pair of entity annotations (e.g. words of the possible event, positions of those words in the sentence, etc.).

Entity annotations present in each sentence are paired and treated as possible event instances. For example, consider the sentence “*In contrast to gene A, the gene B is essential for growth.*” (Fig. 2), in which “*gene A*”, “*gene B*” and “*essential*” are the annotated entities obtained from the NER module. These

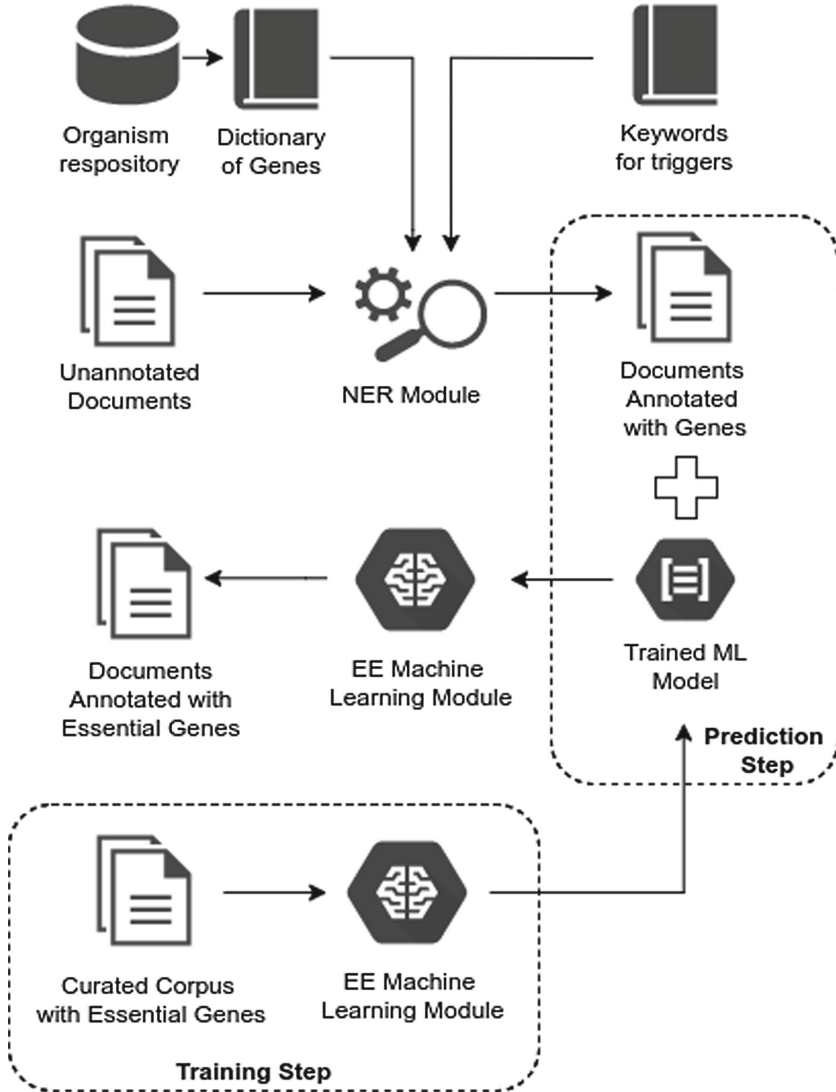


Fig. 1. Overall event extraction pipeline.

annotations are converted into three instances of the matrix considering all possible pairs: “*gene A - gene B*”, “*gene A - essentiality*” and “*gene B - essentiality*” (as observed, the system considers “*gene A - gene B*” as equal to “*gene B - gene A*”).

The EE pipeline contains a filtering approach to select which entity pairs will be used to train ML models. The filtering approach selects which possible pairs of entity classes are allowed in the model (e.g. only pairs that have a trigger and

Table 1. Features used in event classification organized by groups

Feature group	Feature name	Feature description
Sentence morphology	Count tokens between	Counts the number of tokens between two annotations on the event
	Count tokens outside	Counts the number of tokens outside two annotations on the event
	Event annotation between commas	Tests if the annotation event is between commas
	Event annotation between parenthesis	Tests if the annotation event is between parenthesis
	Keyword between event entities	Tests if “not”, “for”, “by”, “in” or “from” keywords exist between event entity tokens
	Lemmas between event annotations	For each lemma, tests if it is present between event token annotations
	Lemmas after keyword until sentence end	For each lemma, tests if it is present in the sentence from keywords “not”, “for”, “by”, “in” or “from” until the end of the sentence.
	Verb between event entities	For each verb, tests if it exists between event annotation tokens
	Verb outside event entities	For each verb, tests if it exists outside event annotation tokens
	Last verb before event	For each verb, tests if it exists before event annotation tokens
Event morphology	Annotation tokens	For each pair of token annotations, checks if it is in the event
	Positions in sentence	For each pair of token annotations, gives the (position/size tokens) in sentence
	Contains token annotations that starts with negative evidence	Tests if annotation starts with “non” or “un” or “in”
	Annotation lemmas	For each pair of lemmas from token annotations, tests if it is in the event
	Annotation part of speech	For each pair of part-of-speech from token annotations, tests if it is in the event
	Event part of speech representative	For each sequence of part-of-speech from all annotations using dependency parsing label nodes, tests if it is in the event
	Event entity annotation type	For each pair of possible entity types, tests if it is in the event
	Dependency tree distance	Minimum distance in number of arcs to get from one entity to other entity in the event

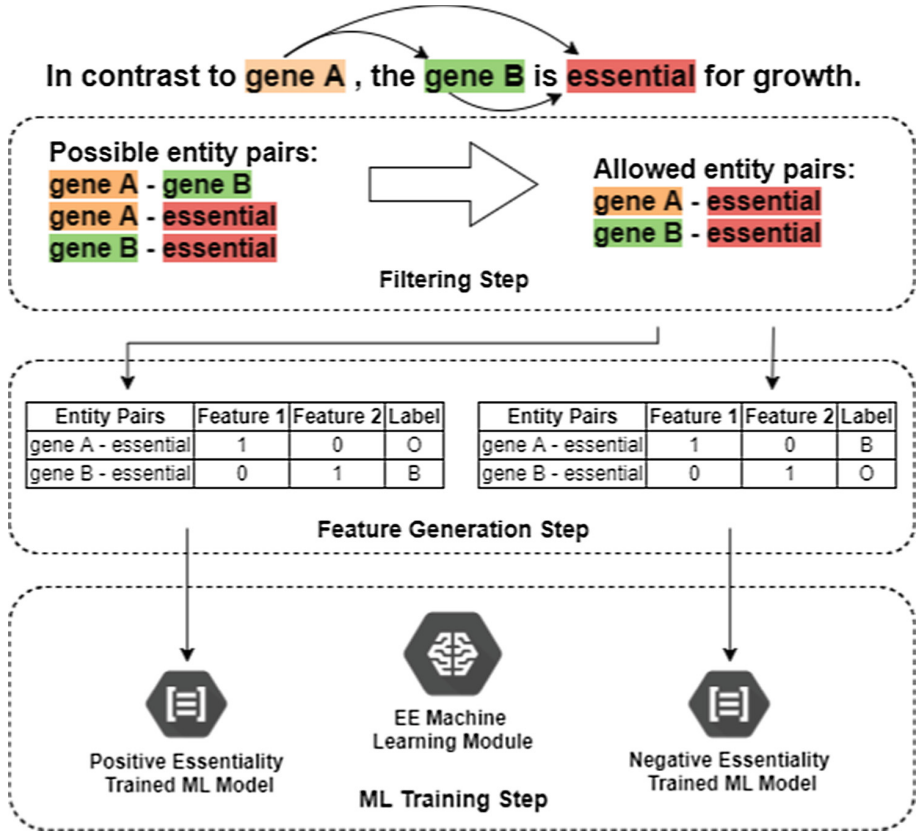


Fig. 2. Sentence example with annotated entities, filtering, feature generation and ML training steps.

gene will be considered in this case). Using the example above, only “*gene A - essentiality*” and “*gene B - essentiality*” are used for the next step.

For each pair selected from the filtering module, a set of features are generated by all modules based on sentence morphology and syntax (Table 1). On the training pipeline, the annotated events are used to label the matrix instances as a binary classification problem:

- “B” for an entity pair that is an event, labelled with the target class of the model;
- “O” for an entity pair that is not an event, or is an event with a type not equal to the target class of the model.

Note that if there are three or more classes in the task, a model for each class is defined. In our case, three models will be trained, for essential, non essential and not related classes.

On the prediction pipeline, each of the ML trained models are applied to a sentence, labeling all pairs of annotations with “B” or “O”, as shown on Fig. 2. Each prediction assigns a score to each possible outcome.

When there are three or more classes, the system allows the combination of several trained ML models that are able to predict events with different classes for the same pair. This leads to an overlap from predictions of two or more models which is solved by selecting the event with the highest prediction score.

The ML algorithm in this work uses Support Vector Machines (SVM), as provided by the LibSVM software library [14]. Default LibSVM parameters are used on training (C-SVC with linear kernel, cost equal to 1 and activated estimation of probabilities).

Evaluation of EE ML trained models can be performed by two evaluation methods: cross-validation or resampling. The evaluation system encompasses a pipeline for dataset processing. The processing pipeline requires the MALLETT framework, that produces the event matrix described above, in which the cross-validation or the resampling evaluations are applied to the event instances.

For cross-validation evaluations, the matrix is split into k folds (k is defined by the user). Each fold is used to predict event annotations from ML model trained on the remaining folds.

For resampling evaluations, the holdout procedure is applied on the matrix. In each evaluation, a percentage of random instances from the matrix is used as the test dataset. The remaining instances are used to train the ML model.

The event extraction system was integrated into the BioTML framework [15] that is available in @Note2 [16], a general-purpose biomedical TM platform <http://anote-project.org/>. This system is accessible through a Java application programming interface (API) that is available at <https://github.com/biotextmining/machinelearning>.

2.2 Manually Curated Corpus for Yeast

An essentiality corpus was created in this work containing 5240 documents related with *S. cerevisiae* randomly selected from PubMed. The corpus was annotated with a dictionary of genes created from the SGD database. Sentences with at least one annotated gene and a trigger like “essentiality”, “essential”, “nonessential”, “non-essential”, “unessential gene” or other gene essential variation keywords were extracted from those documents. Each sentence of the corpus was manually annotated with one of three classes: “essential”, “not essential” or “not related with essentiality” for each pair of gene-trigger annotations.

As a result, 6339 sentences with 6912 gene and trigger annotations were obtained. Those annotations allowed to create 4412 pairs of trigger-gene annotations, which were then classified and separated in 3 main groups (Table 2).

The corpus is also made available at <https://github.com/biotextmining/machinelearning> with annotations in the BioNLP 2011 format (A1/A2 format), allowing to fully reproduce the results of this paper, and to benchmark other methods against the same data.

Table 2. Number of event annotations present in the curated corpus

Event group	Number of annotations
Essential gene	1424
Non essential gene	439
Not related essential gene	2549

3 Results

3.1 Event Extraction System Evaluation

The corpus presented in the previous section was used to validate our EE method. We performed the two possible types of evaluation: cross-validation and resampling.

For cross-validation, events from the whole set of documents were randomly split into k folds. In the results presented in this paper, the value of $k = 10$ was used. The cases in each fold were predicted by an ML model trained using the examples in the remaining folds. The predictions were compared against the manually curated event classes using three metrics: precision, recall and f-score, computed for each of the classes.

For resampling tests, 10 runs of holdout were executed. In each run, $2/3$ of the examples (events) in the corpus were randomly selected to train the ML model. The remaining events ($1/3$) were predicted and compared against the manually curated event classes using the metrics mentioned above. This process was repeated 10 times.

The mean scores obtained from evaluations over the full corpus are described in Tables 3 and 4, for the resampling and cross-validation validations. A confusion matrix from one resampling evaluation run on the full corpus is presented in Table 5.

Table 3. Event extraction resampling evaluation mean scores using the full corpus

Essential gene event group	Recall	Precision	F1
Positive	$56.9 \pm 3.3\%$	$68.9 \pm 2.7\%$	$62.3 \pm 2.6\%$
Negative	$36.5 \pm 8.25\%$	$61.2 \pm 6.4\%$	$45.5 \pm 5.9\%$
Not related	$84.2 \pm 2.1\%$	$72.2 \pm 1.7\%$	$77.8 \pm 1.4\%$

As observed, the trained SVM models performed with similar results in both evaluation methods. The values of f1-scores show better results in the classes with more examples (not related and positive), but a low value on the class with less examples (negative). This is probably due to the original unbalance of the dataset.

Table 4. Event extraction cross-validation evaluation mean scores using the full corpus

Essential gene event group	Recall	Precision	F1
Positive	$57.7 \pm 3.5\%$	$68.8 \pm 7.3\%$	$62.6 \pm 7.50\%$
Negative	$40.7 \pm 7.7\%$	$61.3 \pm 18.2\%$	$48.7 \pm 11.4\%$
Not related	$83.8 \pm 3.5\%$	$72.9 \pm 4.4\%$	$78.0 \pm 3.2\%$

Table 5. Confusion matrix of one run of resampling of full corpus

		Predicted event instances by group event class		
		Not Related Essentiality	Negative Essentiality	Positive Essentiality
Correct event instances by group of event class	Not Related Essentiality	728	27	100
	Negative Essentiality	68	59	18
	Positive Essentiality	203	4	255
Model Accuracy		71.2%		

Overall, these models achieved 71% of accuracy, confirming these suspicions. ML algorithms are highly influenced by the dataset in which they are trained. The results shown that the SVM model was able to predict with high recall and precision on events not related with essentially, because those instances were more frequent than the other two types of events. The ML model predicted negative events with low recall, because the number of examples present in the training corpus was low.

To try to test the results in a balanced dataset, we merged the classes of non-related and negative events. Also, we removed part of the examples in the non related class. As a result, a filtered dataset was created, with 1596 sentences, 3941 entities, 1424 positive events and 1468 not-related/ negative events.

The mean scores obtained in evaluations performed on the filtered corpus are described in Tables 6 and 7. A confusion matrix from one resampling evaluation run on the filtered corpus is described on Table 8.

Table 6. Event extraction resampling evaluation mean scores using the filtered corpus

Essential gene event group	Recall	Precision	F1
Positive	$84.8 \pm 4.3\%$	$76.7 \pm 3.7\%$	$71.7 \pm 2.3\%$
Not related	$68.3 \pm 4.2\%$	$77.1 \pm 6.1\%$	$71.7 \pm 4.5\%$

Table 7. Event extraction cross-validation evaluation mean scores using the filtered corpus

Essential gene event group	Recall	Precision	F1
Positive	82.4 ± 14.5%	76.8 ± 6.9%	79.5 ± 7.9%
Not related	68.0 ± 14, 2%	76.3 ± 10.3%	71.6 ± 8.7%

Table 8. Confusion matrix of a run of resampling of the filtered corpus

		Predicted event instances by group event class	
		Not Related Essentiality	Positive Essentiality
Correct event instances by group of event class	Not Related Essentiality	265	95
	Positive Essentiality	71	406
Model Accuracy		80.2%	

Results show that the corpus balancing resulted in a improvement of overall prediction scores of the ML models, since the algorithm and the parameters used were the same of the previous evaluations.

Positive essentiality events achieved better recall and precision results. This means that our models can, with a high confidence, find sentences that mention essential genes, with precision, recall and accuracy all with values around 80%. This means that ML models can be used to greatly reduce manual curation efforts for this task.

On the other hand, the negative events are harder to discover and to distinguish from non related events. The ability to discover negative events, i.e. sentences were genes are considered non essential, needs trained models with a larger number of examples.

4 Conclusions and Further Work

In this work, we developed an event extraction method, and its implementation in a computational pipeline, that is designed to identify gene annotations and classify them in terms of essentiality for a specific organism. This system is made available in the BioTML plugin, part of @Note2 text mining framework. A manually curated corpus with gene essentiality data was created and made available in this work, allowing to benchmark our method, but also to validate methods proposed by the research community.

The results show that the ML algorithm is highly dependent of the training dataset, because training a ML model with an unbalanced dataset lead to

relatively poor prediction results, mainly in the negative class (events mentioning non essential genes). Balancing the dataset, and considering two classes, an ML model is able to predict with 80% of accuracy and with more than 75% of precision and recall for the positive events.

Deep learning algorithms applied for text mining approaches are emerging with better results than conventional supervised machine learning methods. Further event extraction system improvements could be lead by implementation of deep learning algorithms like long short-term memory networks to automate the feature selection or to identify sentence patterns specific for essential gene identification. Still, the scenario of data scarcity for this task makes this approach more difficult.

Acknowledgments. This work is co-funded by the North Portugal Regional Operational Programme, under the “Portugal 2020”, through the European Regional Development Fund (ERDF), within project SISBI- Ref^a NORTE-01-0247-FEDER-003381.

The Centre of Biological Engineering (CEB), University of Minho, sponsored all computational hardware and software required for this work.

Conflict of Interest. The authors declare they have no conflict of interests regarding this article.

References

1. Guo, D., Zhang, L., Pan, H., Li, X.: Metabolic engineering of *Escherichia coli* for production of 2-Phenylethylacetate from L-phenylalanine. *MicrobiologyOpen* **6**(4), e00486 (2017)
2. Yu, T., Zhou, Y.J., Wenning, L., Liu, Q., Krivoruchko, A., Siewers, V., Nielsen, J., David, F.: Metabolic engineering of *Saccharomyces cerevisiae* for production of very long chain fatty acid-derived chemicals. *Nat. Commun.* **8**, 15587 (2017)
3. Chen, W.H., Lu, G., Chen, X., Zhao, X.M., Bork, P.: OGEE v2: an update of the online gene essentiality database with special focus on differentially essential genes in human cancer cell lines. *Nucleic Acids Res.* **45**(D1), D940–D944 (2017)
4. Cherry, J.M., Hong, E.L., Amundsen, C., Balakrishnan, R., Binkley, G., Chan, E.T., Christie, K.R., Costanzo, M.C., Dwight, S.S., Engel, S.R., Fisk, D.G., Hirschman, J.E., Hitz, B.C., Karra, K., Krieger, C.J., Miyasato, S.R., Nash, R.S., Park, J., Skrzypek, M.S., Simison, M., Weng, S., Wong, E.D.: *Saccharomyces* genome database: the genomics resource of budding yeast. *Nucleic Acids Res.* **40**(Database issue), D700–D705 (2012)
5. Shatkay, H., Craven, M.: *Mining the Biomedical Literature*. Computational Molecular Biology. MIT Press, Cambridge (2012)
6. Gerner, M., Nenadic, G., Bergman, C.M.: LINNAEUS: a species name identification system for biomedical literature. *BMC Bioinform.* **11**(1), 85 (2010)
7. Gooch, P.: BADREX: In situ expansion and coreference of biomedical abbreviations using dynamic regular expressions. *CoRR abs/1206.4*, p. 6 (2012)
8. Campos, D., Matos, S., Oliveira, J.: A modular framework for biomedical concept recognition. *BMC Bioinform.* **14**(1), 281 (2013)
9. Ananiadou, S., Pyysalo, S., Tsujii, J., Kell, D.B.: Event extraction for systems biology by text mining the literature. *Trends Biotechnol.* **28**(7), 381–390 (2010)

10. Yakushiji, A., Tateisi, Y., Miyao, Y., Tsujii, J.: Event extraction from biomedical papers using a full parser. In: Pacific Symposium on Biocomputing, pp. 408–419 (2001)
11. McClosky, D., Surdeanu, M., Manning, C.D.: Event extraction as dependency parsing. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1, HLT 2011, Stroudsburg, PA, USA, pp. 1626–1635. Association for Computational Linguistics (2011)
12. Chun, H., Hwang, Y., Rim, H.-C.: Unsupervised event extraction from biomedical literature using co-occurrence information and basic patterns. In: Su, K.-Y., Tsujii, J., Lee, J.-H., Kwong, O.Y. (eds.) IJCNLP 2004. LNCS (LNAI), vol. 3248, pp. 777–786. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30211-7_83
13. McCallum, A.K.: MALLET: a machine learning for language toolkit (2002). <http://mallet.cs.umass.edu>
14. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. **2**, 27:1–27:27 (2011). <http://www.csie.ntu.edu.tw/~cjlin/libsvm>
15. Rodrigues, R., Costa, H., Rocha, M.: Development of a machine learning framework for biomedical text mining. In: Saberi Mohamad, M., Rocha, M., Fdez-Riverola, F., Domínguez Mayo, F., De Paz, J. (eds.) PACBB 2016. AISC, vol. 477, pp. 41–49. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40126-3_5
16. Lourenço, A., Carreira, R., Carneiro, S., Maia, P., Glez-Peña, D., Fdez-Riverola, F., Ferreira, E.C., Rocha, I., Rocha, M.: @Note: A workbench for Biomedical Text Mining. J. Biomed. Inform. **42**(4), 710–720 (2009)



Rise, Fall, and Implications of the New York City Medallion Market

Sherraina Song^(✉)

Shrewsbury High School, Shrewsbury, MA 01545, USA
sherraina.s@gmail.com

Abstract. Capping the number of licenses and granting exclusive right to street hailing passengers, the New York City (NYC) medallion system manipulated the demand and supply of taxicab services and created a medallion market. The lasting system turned the right to operate taxis in NYC into a private property of scarcity and an investment vehicle with disguised risks. Integrating data published by the NYC Taxi and Limousine Commission (TLC), this research identified four phases of the medallion market and argued that (1) the market collapsed because technology and ride-sharing economy have materially weakened the assumptions underlying the market; (2) Yellow Cab is fighting a lost battle against players of ride-sharing economy; and (3) the deregulation of the NYC taxicab industry will adapt and continue despite its adverse impact on the medallion interest groups.

Keywords: App-based · Boro Taxis · For-Hire Vehicle (FHV) · Green Cab · Haas Act · Medallion · New York City (NYC) · Street Hail Livery (SHL) · Ride-sharing · Taxicab · Taxi and Limousine Commission (TLC) · Uber · Yellow Cab

1 Introduction

1.1 NYC Taxicab Market

The NYC taxicab market is one of the largest in the world, with about one million passengers per day and annual revenue of two billion US dollars. By the local government regulations, summarized in Table 1. Classification of NYC Taxicab Services and Providers, the market consists of two sectors of service demand (street hailing and pre-arranged pick-up) and three major classes of service suppliers: Yellow Taxi Cab (Yellow Cab), For-Hire Vehicles (FHV), and Street Hail Livery (SHL).

Street hailing services are provided by taxicabs in response to hails by passengers on the streets. Pre-arranged pick-up services are provided by taxicabs in response to requests made to a taxicab's affiliated service dispatching base.

Identifiable by the color of canary yellow, Yellow Cab taxis are providers of street hailing services. They are permitted to pick up passengers anywhere in all the five NYC boroughs. More, they are granted exclusive right to street hailing in Manhattan, LaGuardia Airport, and John F. Kennedy International Airport [1], where most of the traditional NYC taxi traffic is originated or destined. Customers access this mode of

transportation by standing in the street and hailing with hands. A medallion, the metal plate attached to a car's hood, is the proof of legal license, i.e., the right for a car to provide street hailing services. There is a cap on the number of available licenses.

Table 1. Classification of NYC taxicab services and providers

	Right to street hailing passengers	Right to pre-arranged pick-ups
Yellow Cab	All NYC	Not permitted
FHVs	Not permitted	All NYC
Green Cab	Northern Manhattan (north of West 110th street and East 96th street) and outer-boroughs (Bronx and Queens excluding the airports)	All NYC

FHVs include Community Cars (aka Liveries), Black Cars, and Luxury Limousines. Those taxicabs can pick up passengers throughout the five NYC boroughs, but only by appointments [2]. Customers access this mode of transportation by submitting a request, via phone, mobile apps, website, or other methods, to a TLC-licensed base or a TLC-licensed dispatch service provider who then direct FHV taxicabs to the customers. Important to note, app-based service providers such as Uber and Lyft are classified as FHVs. They were not permitted to enter the NYC taxicab market until the middle 2011. However, once permitted, they became disruptive against street hailing service providers as smart phones made FHVs as convenient (if not more so) as traditional taxicabs.

SHL, painted apple green and known as “Boro Taxis” or “Green Cabs”, is a hybrid between Yellow Cab taxis and FHVs. They are permitted to accept pre-arranged rides in all the five NYC boroughs, and, beginning in June 2013, are permitted to pick up hailing passengers from the street in northern Manhattan (north of West 110th street and East 96th street) and the outer-boroughs: the Bronx and Queens (excluding the airports), areas historically underserved by Yellow Cab [3].

1.2 Medallion System

Licenses for both drivers and vehicles are required to operate taxicabs in NYC. However, regulations vary on Yellow Cabs, Green Cabs, and FHVs. Yellow Cabs have the strictest licensing. The right to serve street hailers had been exclusively assigned to Yellow Cabs until 2013 – the year Green Cab was created. The number of Yellow Cab Taxis permitted on streets is controlled via medallion licensing, by which the New York state’s legislative body approves additional medallions and the NYC TLC holds auctions to sell them to public.

Introduced in 1937, the medallion system added only limited number of Yellow Cabs with only three legislative approvals in year 1996, 2004, and 2013. Today, only 13,587 Yellow Cab taxis are permitted on the NYC streets, corresponding to the same number of medallions.

2 Evolution of NYC Medallion Market

Based on the data integrated from the TLC websites, annual transfer volumes and average prices for Yellow Cab medallion transactions are graphed in Fig. 1. NYC Yellow Cab Medallion Annual Transfers and Average Sales Prices. According to the price and volume movements, the market can be described in four different phases,

- Born but no value (1937–1946)
- Formed and established (1947–1986)
- Investment tool and booming (1987–2013)
- Collapsed and falling (2014 – Present).

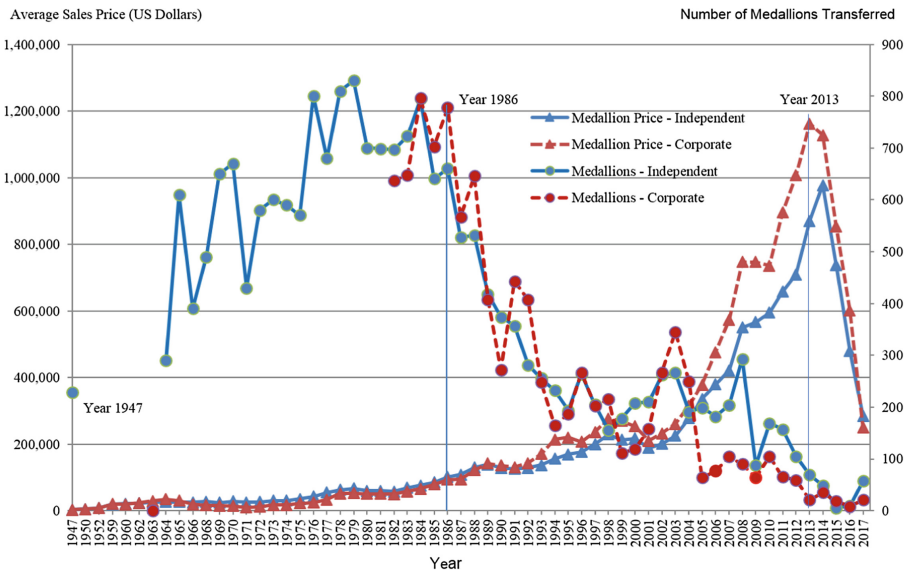


Fig. 1. NYC Yellow Cab medallion annual transfers and average sales prices

Data details are in Table 2. NYC Yellow Cab Medallion Annual Issuance and Sales Transfer.

2.1 Born But No Value (1937–1946)

The first phase of nine years from 1937 through 1946 carried no market values for the medallions. After the great depressions in early 1930s, NYC was flooded with 30,000 drivers. Sometimes there were more taxi cabs than passengers on the streets. Out of concerns about congestion, pollution, and crimes, the Haas Act was legislated in 1937 and official taxis were introduced with medallions attached. The law limited the number of cab licenses to the existing 16,900, but only 13,595 were in active use due to registration lapses [4]. The active licenses dwindled to 11,787 in 1947 due to reduced renewals and stood the same for 50 years until 1996 when 266 more were issued.

Table 2. NYC Yellow Cab medallion annual issuance and sales transfer

Medallion market	Year	Total medallions				Independent medallions		Corporate medallions	
		Approved	Issued	Active	Sales transfer	Average price (\$)	Numbers traded	Average price (\$)	Numbers traded
Phase I	1937–1946	16,900	13,595	13,595	0	0	0	0	
Phase II (1947– 1986)	1947					2,500		2,500	
	1950					5,000		5,000	
	1952					7,500		7,500	
	1959					19,500		20,000	
	1960					20,825		19,450	
	1962					22,000		23,400	
	1963					25,000		28,773	
	1964			11,787	290	26,000	290	34,145	
	1965			11,787	610	26,000	610	30,000	
	1966			11,787	390	25,000	390	19,000	
	1968			11,787	490	27,000	490	16,000	
	1969			11,787	650	24,500	650	15,000	
	1970			11,787	670	28,000	670	14,000	
	1971			11,787	430	25,000	430	10,000	
	1972			11,787	580	26,000	580	12,000	
	1973			11,787	600	30,000	600	17,000	
	1974			11,787	590	30,000	590	17,000	
	1975			11,787	570	35,000	570	22,000	
	1976			11,787	800	42,000	800	24,000	
	1977			11,787	680	55,000	680	33,000	
1978			11,787	810	63,000	810	52,000		
1979			11,787	830	67,000	830	53,000		
1980			11,787	700	60,000	700	50,000		
1981			11,787	699	60,000	699	50,000		
1982			11,787	1,334	57,500	697	49,300	637	
1983			11,787	1,371	68,600	723	57,900	648	
1984			11,787	1,591	75,900	795	66,200	796	
1985			11,787	1,344	84,900	641	79,000	703	
1986			11,787	1,438	101,600	660	92,900	778	
Phase III (1987– 2013)	1987			11,787	1,094	108,700	527	94,600	567
	1988			11,787	1,178	129,700	532	121,500	646
	1989			11,787	826	139,100	418	141,400	408
	1990			11,787	646	128,400	374	135,700	272
	1991			11,787	800	126,067	357	130,360	443
	1992			11,787	688	128,577	281	143,199	407
	1993			11,787	504	137,196	256	170,200	248
	1994			11,787	396	155,633	232	214,221	164
	1995			11,787	381	169,750	194	219,958	187
	1996	400	266	12,053	531	176,333	264	207,292	267
	1997		134	12,187	408	199,875	205	236,500	203
	1998			12,187	370	229,000	155	277,318	215
	1999			12,187	289	212,917	178	269,500	111

(continued)

Table 2. (continued)

Medallion market	Year	Total medallions				Independent medallions		Corporate medallions	
		Approved	Issued	Active	Sales transfer	Average price (\$)	Numbers traded	Average price (\$)	Numbers traded
	2000			12,187	327	217,125	208	253,864	119
	2001			12,187	368	188,958	210	209,458	158
	2002			12,187	529	200,333	262	232,250	267
	2003			12,187	611	224,958	266	260,917	345
	2004	1,050	554	12,741	440	277,583	191	315,636	249
	2005		38	12,779	263	335,583	199	378,556	64
	2006		249	13,028	259	379,000	182	476,000	77
	2007		120	13,148	308	420,964	204	573,489	104
	2008		89	13,237	383	550,000	293	747,000	90
	2009			13,237	150	566,732	87	746,746	63
	2010			13,237	274	595,118	169	736,200	105
	2011			13,237	222	657,665	157	895,462	65
	2012			13,237	164	709,643	105	1,007,203	59
	2013	2,000	200	13,437	90	870,059	69	1,162,381	21
Phase IV (2014-)	2014		150	13,587	84	977,729	49	1,127,371	35
	2015			13,587	24	736,667	6	852,500	18
	2016			13,587	17	479,191	9	600,266	8
	2017			13,587	78	285,168	57	249,891	21

The number-capped medallion system had little impact on the NYC taxi industry during this initial phase. Due to the World War II and lack of demand for taxi services, many medallion owners valued a medallion not worth the annual \$10 renewal fee and chose not to renew. No evidences suggest that, in capping the number of taxis, the law makers of Haas Act intended to turn the right to operate taxicabs on the NYC streets into a property with tradable market value. The establishment of medallion was not much different from other Depression-era legislative efforts: to stabilize and revive the taxicab industry diagnosed suffering from excessive competition [5].

However, the Haas Act did have an ordinance allowing transfer of licenses between owners, conditionally upon the NYC's approval of new owners' qualifications. This transferability was critical to establish medallion values and trade in future when economic conditions improved and demand for taxicabs rose.

2.2 Formed and Established

A medallion market was formed and stabilized during the second phase of almost four decades from 1947 through 1986. Until 1947 had there been no demand adequate to utilize the existing medallions from individuals seeking to drive a taxi. Rationing of fuel and car parts during World War II turned more people to taxis for transportation and the post-World War II prosperity created more business, which led to more drivers than the medallions available [6]. Medallions started to assume value and a medallion market formed in response to the need of medallion trading.

In 1947, the New York Times reported that taxicab owners received bonuses averaging \$1,500 or \$2,500 from selling their medallions with used cabs [7]. In 1950, the “bonus” rose to \$5,000. The “bonus”, on the top of the sales price for a cab, effectively put a price tag on a medallion and indicated the birth of a standalone market. In early 1960s, a medallion was traded around \$25,000.

In 1971, the NYC TLC was created pursuant to Local Law 12 of 1971 to license taxicab vehicles and drivers by establishing and enforcing standards and criteria [8]. The creation and functioning of the TLC brought regulation transparency and consistency, which contributed to the health and stability of the NYC taxicab industry. It also led to legitimization of “gypsy cabs” into what known today as livery cars, community cars, car services or for-hire vehicles [9]. In its annual reports, the TLC stated “taxicab licenses are transferable, and may be pledged as security for loans. . . ., the license has a considerable value” [10]. Explicitly, the TLC pointed to the tradable value of the medallion and existence of a medallion market.

In addition to purchasing that requires a large amount of payment up front, drivers can pay medallion owners for the right of use by operation shift or certain hours, i.e., leasing. The Haas Act mandated that 60% of the medallions go to fleets who hold two or more licenses and can rent them to drivers. In 1979, TLC legalized leasing. Through waves of conversion to lessee-driving from owner-driving - fleet leasing in 1980s and independent owner leasing in 1990s, nearly all fleet drivers and most independent drivers were lessees [11]. Medallion ownerships were effectively separated from their right of use, which made it easy to price and trade medallions. Independent agencies were founded to broker and manage leasing on behalf of medallion owners.

During the second phase, demand for taxicabs rose due to a growing NYC population – residents and tourists, most of whom did not drive, while the number of medallions was capped the same for the whole period. Sales transfer of medallions increased at steadily higher prices. In 1984, the trading volumes went as high as 1,591, or 13.5% of the medallions in circulation. In 1986, 1,438 medallions or 12.2% of existing medallions changed hands. Thereafter both the number of medallions traded and its percentage in the total continued to decrease. As such, year 1986 was deemed the end of this forming phase. In the same year the average sales price for a medallion crossed \$100,000 for the first time, forty times the price at the beginning of this phase (inflation was not adjusted). Without new supply, this phase of the NYC medallion market was characteristic of more transactions, rising prices, and establishment of a regulation agency, the NYC TLC.

2.3 Investment Tool and Booming

During the third phase of 1987 through 2013 (or quarter 2 of 2014 concisely), the medallion price continuously rose, but trading volumes were thin, and thinner. This trend continued despite additional 1,650 or 14% more licenses were issued between 1996 and 2013. Both private sales transfers and official auctions kept recording prices historically high. Medallions were bought and held in anticipation for value appreciation. The license became an investment vehicle, no longer limited to the way gaining the right to drive to make a living in the city. Medallion-owner drivers populously counted on selling their medallions later to make comfortable retirements.

The annually averaged prices peaked at \$1.16 million for a corporate medallion in 2013 and \$0.98 million for an independent one in 2014. Only 90 medallions were transacted in 2013, 21 corporate and 69 independent, less than 0.67% of the total in use. The price was so high that fractions of a medallion were recorded in sales transfer. Except temporary setbacks from the economic recessions in early 1990s and following the terrorist attack on September 11 of 2001, the price trend line was straight up, projecting the medallion as a safe investment risking nothing; the volume trend line was straight down, telling few owners would like to sell. The two lines crossed and formed the shape like a pair of scissors in Fig. 1.

Rising price and known entry control made the NYC medallion a safe bet and gave it many attributes of the bundle of right as a private property [12]. It has been routinely bought and sold, leased, and used as collateral for loans and counted as assets in estate, bankruptcy, divorce, and inheritance settlements. Purchases were financed through credit unions, banks, and other financial institutes. In 1995, Medallion Financial was founded as a firm specialized in originating, acquiring, and servicing loans that finance taxicab medallions and various derivatives. It was listed and actively traded one year later in NASDAQ stock exchange [13]. Taking the Schaller Consulting's estimate that 15% of a medallion's total revenues went to its owners [14], the annual return was computed between 3% and 9% during this period, better than investment in gold and oil in the comparable years. Calculated according to the rate rules in the TLC Promulgation of Rules issued in 2012 [15], a medallion owner can earn \$30,000 to \$80,000 annually by leasing out one medallion. It was commonly believed that buying, holding, and leasing out medallions was a wise business decision. The environment of low interest rate following 2008 financial crisis provided widely accessible, low-cost loans and contributed to the medallion hype as well.

Medallion auctions administrated by the TLC also enforced the perception that investing in the medallion was safe. "Strong medallion sale prices have historically been used to judge the overall health and viability of the industry" [16] was frequently presented in the TLC annual reports. It was no coincidence that TLC auctions always set price records.

2.4 Collapsed and Falling

The good time peaked and started to end in the second half of 2014. Viewed quarterly, the average sales price for a corporate medallion peaked at \$1.26 million in quarter 2 of 2014 and then fell straight to \$208,411 in quarter 4 of 2017, a drop of 83.5%; the average sales price for an independent medallion peaked at \$1.0 million in quarter 3 of 2014 and then fell straight to \$191,749 in Quarter 3 of 2017, a drop of 80.8% (Fig. 2. NYC Yellow Cab Medallion Quarterly Sales Prices). The fall was steep and fast. It took almost twenty years to rise to \$1 million for a medallion in 2013, but less than three years to fall back where it was: around \$200,000. Quarterly data details for individual years are in Table 3. NYC Yellow Cab Medallion Quarterly Sales and Prices.

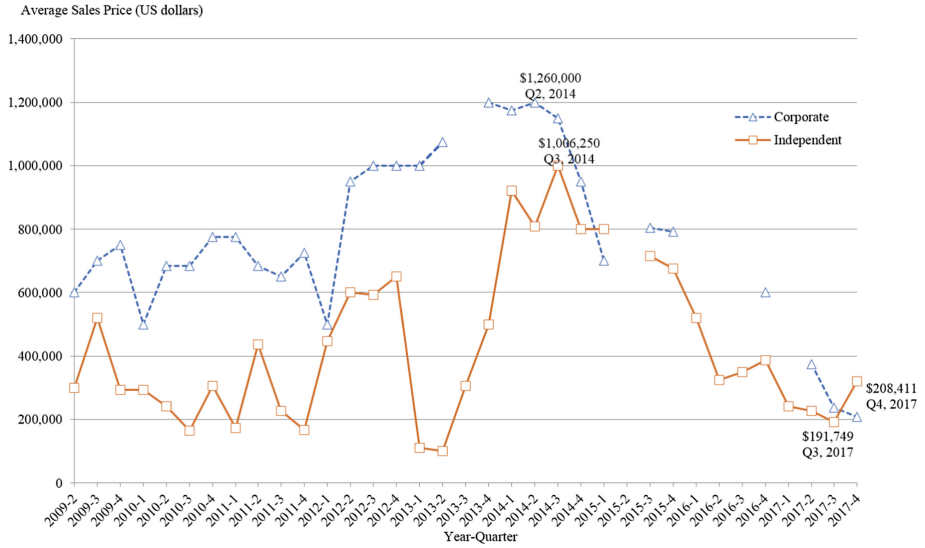


Fig. 2. NYC Yellow Cab medallion quarterly sales prices

Table 3. NYC Yellow Cab medallion quarterly sales and prices

Year	Quarter	Corporate							Independent				
		Transactions	Medallions		Price ('000 US dollars)			Transactions	Medallions		Price ('000 US dollars)		
			Transacted	Transferred	High	Low	Average		Transacted	Transferred	High	Low	Average
2009	2	13	27	27	763	600	735	28	28	27	578	300	561
2009	3	16	32	32	775	700	755	52	52	49	594	520	572
2009	4	2	4	4	775	750	763	11	11	11	600	293	557
2010	1	14	29	29	800	500	721	45	45	39	620	293	577
2010	2	18	36	36	800	685	741	43	43	39	610	242	589
2010	3	13	27	27	825	685	713	51	51	44	615	165	596
2010	4	6	13	13	850	775	807	53	53	48	700	305	614
2011	1	5	10	10	950	775	915	37	37	31	660	173	608
2011	2	9	18	18	975	685	878	74	74	58	695	438	661
2011	3	9	19	19	950	650	862	43	43	35	705	227	675
2011	4	8	18	18	1,000	725	938	37	37	35	710	167	679
2012	1	5	10	10	1,000	500	890	25	25	24	715	447	683
2012	2	6	12	12	1,050	950	996	38	38	35	712	600	703
2012	3	11	22	22	1,125	1,000	1,039	35	35	31	750	592	708
2012	4	7	15	15	1,125	1,000	1,048	16	16	16	850	650	769
2013	1	3	6	6	1,210	1,000	1,103	19	19	16	950	112	822
2013	2	3	6	6	1,320	1,075	1,165	26	26	26	1,100	100	914
2013	3				0	0	0	13	13	13	1,050	305	833
2013	4	4	9	9	1,200	1,200	1,200	14	14	14	1,000	499	878
2014	1	6	12	12	1,254	1,175	1,205	25	25	25	1,050	920	985
2014	2	2	5	5	1,300	1,200	1,260	17	17	15	1,050	808	1,002
2014	3	3	6	6	1,200	1,150	1,183	4	4	4	1,025	1,000	1,006
2014	4	6	12	12	1,000	950	967	5	5	4	866	800	827
2015	1	4	8	8	950	700	863	2	2	2	800	800	800
2015	3	2	4	4	875	805	840	3	3	3	715	715	715
2015	4	3	6	6	875	793	848	1	1	1	675	675	675
2016	1				0	0	0	2	2	2	580	520	550
2016	2				0	0	0	4	4	4	600	325	476
2016	3				0	0	0	2	2	2	570	350	460
2016	4	4	8	8	675	475	600	1	1	1	388	388	388
2017	1				0	0	0	1	1	1	241	241	241
2017	2	1	2	2	375	375	375	8	8	8	300	150	228
2017	3	10	19	19	140	236	236	9	9	9	256	130	192
2017	4	1	1	1	208	208	208	39	39	39	628	150	320

Table 4. NYC Yellow Cab medallion foreclosures

Year	Quarter	Corporate					Independent				
		Foreclosures	Medallions	Recorded unit value ('000 US dollars)			Foreclosures	Medallions	Recorded unit value ('000 US doallars)		
				Foreclosed	Highest	Lowest			Average	Foreclosed	Highest
2011	3						1	1	635	635	635
2014	3						1	1	900	900	900
2014	4	1	1	1,925	1,925	1,925	3	3	905	840	873
2015	1						1	1	800	800	800
2015	2						3	3	777	700	745
2015	3						3	3	725	603	681
2015	4						3	3	725	326	585
2016	1										
2016	2						7	7	615	540	574
2016	3	10	10	1,500	1,250	1,325	5	5	620	550	602
2016	4						3	3	600	550	583
2017	1						1	1	550	550	550
2017	2	1	2	738	738	369	7	9	500	220	348
2017	3	1	2	202	202	202	17	20	581	185	420
2017	4						8	13	450	200	354

Not only were the prices low, but also the transaction volumes were light. Unlike the third phase with few sellers due to appreciation expectations, the fourth phase had fewer sellers because of no buyers when no retainable floor prices were in sight. Corporate and independent medallions combined, only 24 changed hands in 2015 and 17 in 2016, out of the total 13,587. Together, those transactions made a sale of only \$20 million in 2015 and less than \$9 million in 2016. The market, valued over \$14 billion prior to quarter 4 of 2014, collapsed.

Without buyers, many owners were unable to pay back their loans and filed for bankruptcy. Between quarter 3 of 2014 and quarter 4 of 2017, 15 corporate medallions were foreclosed, defaulting loans valued over \$16.3 million, averaged \$1.09 million per piece; 72 independent medallions were foreclosed, defaulting \$35.8 million, averaged about \$0.5 million each. In contrast, there was only one foreclosure recorded (in 2011) prior to quarter 3 of 2014 (Table 4 NYC Yellow Cab Medallion Foreclosures). Not a surprise, impact on independent medallions owned by drivers is far more severe than that on those medallions owned by corporates.

Many medallions are now in possession of credit unions and banks who financed the purchases. In the middle September of 2017, for a total of \$8.56 million, or \$186,000 per medallion, a hedge fund company won the auction sale of 45 medallions foreclosed from an owner who once owned 800 medallions [17]. More foreclosures are likely to follow. Aware of the market distress, the TLC had to hold off auctioning the remaining 1,650 of the 2,000 Yellow Cab medallions authorized in 2013.

3 The Uber Disruption

Blames have been quickly played against Uber for the meltdown of the NYC medallion market – less regulated than Yellow Cab and thus gained an edge in competition.

Table 5. NYC taxicab ridership and market share by provider

Year-quarter	Total market trips (million)	Share of total market trips (%)				
		Yellow Cab	Green Cab	FHV's		
				Uber	Lyft	Other FHV's
2015-1	52.8	73.01	9.10	12.51	0.02	5.37
2015-2	55.1	69.99	9.24	12.69	0.54	7.54
2015-3	55.3	61.39	8.27	18.65	1.75	9.95
2015-4	64.7	54.26	7.38	19.16	2.09	17.11
Year total	227.8	64.14	8.44	15.93	1.15	10.34
2016-1	67.1	50.85	6.68	21.57	2.96	17.94
2016-2	71.2	49.04	6.30	22.23	4.05	18.37
2016-3	67.9	44.71	5.51	26.77	4.55	18.46
2016-4	73.7	42.65	4.92	29.33	4.69	18.41
Year total	279.8	46.74	5.84	25.04	4.08	18.30
2017-1	75.6	38.59	4.30	31.62	6.48	19.01
2017-2	53.5	37.64	4.00	32.94	7.18	18.24
Year to date	129.1	38.19	4.17	32.17	6.77	18.69

Uber and Lyft have been winning both market shares and revenues. Beginning for year 2015, the TLC published trip data for all the providers - Yellow Cab, Green Cab, and FHV's, which made it possible to view individual providers' market shares (Table 5 NYC Taxicab Ridership and Market Share by Provider). By quarter 2 of 2016 Yellow Cab had retreated to take less than 50% of the NYC taxicab service market. While the quarterly market size increased to 75.6 million trips in the first quarter of 2017 with a growth of 43% over the same period two years ago, Yellow Cab's trips dropped to 29.2 million, a loss of 9.4 million or 24%; and its market share dropped to 38.6% from 73%. Not only failed Yellow Cab in grabbing a share from the market growth but also it failed in customer retention – many riders who used to hail a Yellow Cab taxi now turned to Uber, Lyft, or other small FHV's. In the first quarter of 2017, FHV's as one group served 57.1% of the NYC taxicab trips and became NYC riders' first choice. Uber alone captured 31.6%, up from 12.5% two years ago. Figure 3 Market Share of NYC Taxicab Ridership by Provider illustrated the wining stride of Uber and Lyft, in contrast with Yellow Cab's drastic retreat, quarter after quarter.

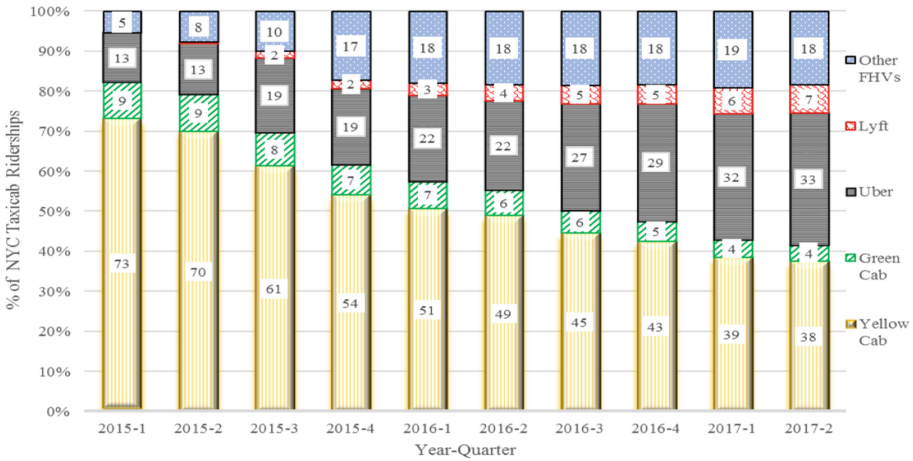


Fig. 3. Market share of NYC taxicab ridership by provider

Table 6. NYC Yellow Cab annual trips and revenues

Year	Medallions	Trips			Revenues		
		Total	Per medallion	Year over year change %	Total (\$)	Per medallion (\$)	Year over year change %
2010	13,237	168,983,489	12,766		1,789,049,841	135,155	
2011	13,237	176,866,900	13,362	4.67	1,992,549,043	150,529	11.37
2012	13,237	177,996,949	13,447	0.64	2,134,910,742	161,284	7.14
2013	13,437	173,136,240	12,885	-4.18	2,322,802,868	172,866	7.18
2014	13,587	165,104,282	12,152	-5.69	2,268,307,017	166,947	-3.42
2015	13,587	146,107,068	10,753	-11.51	2,097,292,315	154,360	-7.54
2016	13,587	130,789,390	9,626	-10.48	1,906,905,626	140,348	-9.08

Trips and revenues per medallion also dropped. Between 2013 and 2016, Yellow Cab’s annual trips per medallion dropped 25%, to less than 10 thousand from around 13 thousand, and annual revenues per medallion dropped 19%, to \$140 thousand from over \$170 thousand (Table 6 NYC Yellow Cab Trips and Revenues). Taking 15% as medallion owner’s share, an average Yellow Cab medallion earned a return of only \$21,000 in 2016, compared to \$25,900 just three years earlier, and \$30,000 to \$80,000 during its booming period.

4 Economic and Regulation Implications

4.1 Breakdown of Market Assumptions

Market functioning and values of the NYC medallions have been relying on one supply policy - restricting issuance of medallions and thus controlling the number of taxicabs on streets and one demand assumption - street hailing and pre-arrangement are two

different demand for taxicab services and can be met with two different service products. Thus, it goes that government can segment the market and designate different service providers accordingly and exclusively. It was further assumed there be growing number of street hailers out of growing economy, visitors, and residents who prefer not to drive or conscious of traffic congestion, air pollution, and inconvenience in driving, which has been mostly true. As such, the supply holds flat while the demand grows, medallion values and driver revenues are assured to rise – law of economics 101.

However, those assumptions, even if used to be true, have been disrupted by emergence and advance in technology-enabled ride-sharing economy. By 2015, 96% of NYC residents had owned mobile phones and 79% of those were smart phones [18]. Almost all taxicab riders can tap mobile apps or dial up from handset devices to pre-arrange a cab, anywhere and anytime, on streets or off streets. The lagging time between hailing and pre-arranging became no longer significant and meaningful. When there are enough taxicabs nearby waiting for the calls, callers can get the benefit of immediacy and convenience, almost no different (if not better off) from that of hailing Yellow Cab taxis. Even more, people would prefer to call their cabs prior to getting off flights, leaving restaurants and coffee shops, and from places of comfort instead of hailing in cold weather or in the rains. App-based on-demand pre-arrangement offers the benefits of instant planning and predictability. The demand for pre-arranged taxicabs has become hardly differentiable from street hailing, or at least the attributes used to enable riders to differentiate the two have diminished or blurred in riders' eyes. The two have become two units of one product that transports people, substitutable to each other, and should be regulated as one product [19]. The primary assumptions, upon which the NYC medallion market and government regulations have been based and functioning, have been fundamentally uprooted.

4.2 Ride-Sharing Economy

When demand for app-based riding rose and the number of Yellow Cab taxis on the NYC streets were restricted, FHV's responded by adding more vehicles and drivers. There were no FHV's legally permitted on NYC streets when the medallion system was born. But the number of FHV's was more than doubled to 80,881 and the FHV drivers increased by 120% to 122,997 between 2011 (the year Uber entered NYC) and 2016. For the same period, only 300 Yellow Cab taxis were added (Table 7 NYC Active Taxicabs and Drivers). More, 19,463 Yellow Cab drivers quit and most of them switched to FHV's. The net result is that FHV's outnumbered Yellow Cab taxis by 6:1; FHV drivers outnumbered Yellow Cab drivers by 4:1; and up to four drivers had to share driving one Yellow Cab vehicle by shift due to the medallion restriction. Uncapped licensing seemingly did give FHV's advantages over Yellow Cabs under the existing regulations.

Successes in business models like Uber's are not uncommon in the era of digital economy – eBay, Amazon, Facebook, Google, etc. Leveraging Internet and smart phones, they built platforms to connect and assemble buyers and sellers directly to create a market ecosystem, economy of scale, and even monopolies via “Size begets size” [20]. Different for Uber, a pioneer in sharing economy, it explores and exploits resources idle prior to the Internet economy – private cars at the times not being driven and personnel at the times outside regular jobs, which makes it theoretically possible

Table 7. NYC active taxicabs and drivers

Year	Drivers		Vehicles		Vehicle driver ratio	
	Yellow Cab divers	FHV drivers	Yellow Cab medallions	FHVs	Yellow Cab	FHVs
1937			13,595	0		
1964			11,787	2,513		
1992			11,787	27,613		
2000	35,160	48,271	12,187	41,813	2.9	1.2
2005	42,512	51,060	12,779	40,449	3.3	1.3
2010	49,129	53,755	13,237	37,782	3.7	1.4
2015	55,390	90,284	13,587	66,604	4.1	1.4
2016	30,488	122,997	13,587	80,881	2.2	1.5

for almost everyone to become an Uber driver and thus provides options and flexibilities in offering, scheduling, and pricing to compete. As it evolves and adapts to market demand and regulations, new features can be expected to address public concerns the medallion system initially intended to address – traffic congestion, air pollution, and safety. For example, dynamic pricing with surcharges can be explored to contain traffic through crowded areas; access to driver and passenger information and their mutual rating can be explored to improve safety. There are advantages over Yellow taxicab drivers who must earn or lease a medallion up front and adhere to stricter licensing criteria and regulated pricing.

Globalization and market size matters too. An estimated online advertising market of \$1 trillion has created the legendary Google and Facebook. The global market for personal mobility is as much as 10 times that [21]. Appealing to investors with the ambition to be another Google or Facebook, Uber has attracted \$18 billion in funding since its setup in 2010 and now carries a valuation closed to \$70 billion, the largest startup in history that raised the most money even before going public. The large capital enabled Uber to extend its platform and business model to more than 450 cities in 78 countries [22] and to build its fleets of autonomous driving for future. In contrast, the medallion system in NYC or the similar ones in other cities confine their taxicab service providers to a geographic locale, potentially blocking their riders from not only the benefits of sharing economy but also the prospects for Uber or any of its existing or potential competitors to replicate those legendary successes in an era of digital age.

4.3 Deregulation Trend

Consumers are standing to benefit from the ride-sharing economy. In the era of mass intelligence and digital economy, the new service mode of ride-sharing has made taxi riding more accessible and affordable, which helps grow the market. In 2016, total NYC taxicab ridership has got bigger, to 280 million, up 23% from one year ago. Meanwhile Uber and Lyft gained not only from the market incremental but also from what Yellow Cab lost: 5.3 million trips and 17.4% market share during the same period. If not lost to Uber, it would have lost to someone else who can materialize the

benefit. Technology is there, demand is there, and consumers are ready to make moves in their riding and opinions on taxicab regulations. Uber and the likes are in right places at right times. But nothing is assured who will be the eventual winner, facing evolving technology, increasing market competition, and regulations that certainly will adapt.

Though the perpetuation of the medallion system was the result of political process subject to more influence from interested supplier groups – owners, drivers, agencies, and creditors than from consumers, political winds seem to shift toward deregulations favoring Uber and the likes who run their business on national and global scale beyond localized monopolies. The advantage of financial power, easily identified common interest, and ease of organizing the medallion interest groups over insufficiency in funding and difficulty in organizing consumers (whose individual interests in taxicab market are scattered and ambiguous), is among the main reasons why the medallion system was perpetuated and lasting [23]. Now Uber, with sufficient funding, concentrated investor interest, and organization power in influencing law makers and public opinion, is up to the task to challenge the traditional order and medallion interest groups. It mobilized public support and launched political campaigns to change regulations. It started “principled confrontation” program in 2015, searching for compromises with local municipalities for entry into their markets. In the summer of 2015, Uber won against NYC and foiled the city’s efforts to cap the number of Uber vehicles on the grounds of traffic congestion [24]; in September of 2015, the New York City won a legal victory against three lawsuits brought by Melrose Credit Union, the largest lender who made almost \$2.5 billion in loans for 5,331 city-issued medallions and claimed it was illegal for Uber and other app companies to operate in New York City [25]; In May of 2016, the New York state senate passed bill to legalize Uber statewide [26]. Similar efforts and successes in other places have produced ordinances favorable to app-based services in more than 23 states in the United States.

The deregulation process of the taxicab industry has started and hardly can that be turned back by any foreseeable political winds. After all, the New York medallion is a “problematic private property” - created in the past, controversial in the present, and potentially burdensome in the future [27]. Instead of patching and reviving the medallion system, local and federal regulations should adapt and, progressively but decisively, catch up with technological innovations and changes in consumer demand. The essence is to let the free market play freely and let the once protected medallion monopoly adapt or die. Instead of holding Uber and the likes back, regulations will foster their growth, monitor their expansions, and intervene timely to prevent them from propelling into monopoly powers like Google, Facebook, and Amazon.

Source Data

1. **TLC Annual Reports (2002–2016)**. <http://www.nyc.gov/html/tlc/html/archive/annual.shtml>. Summarized the TLC work, including licensing and regulation updates.
2. **Medallion Transfer Reports (2009–2017)**. <http://www.nyc.gov/html/tlc/html/archive/archive.shtml>. Click on a link for a year, then the link of Medallion Transfers.

3. **Trip and Revenue Data (2010–2017).** http://www.nyc.gov/html/tlc/html/technology/aggregated_data.shtml. Data for Yellow Cab and Green Cab at monthly level and data for FHV's at weekly level were integrated to derive metrics of taxicab trips and revenues.

Acknowledgements. The author would like to thank the three anonymous reviewers for their encouragement and comments that helped me greatly improve the manuscript. I am immensely grateful to Kaitlin Andryauskad for her mentorship, inspirations, and insights that motivated and helped me to develop the project and finalize this paper.

References

1. Yellow Taxi. http://www.nyc.gov/html/tlc/html/industry/yellow_taxi.shtml. Accessed 16 Apr 2018
2. For-Hire Vehicles. http://www.nyc.gov/html/tlc/html/industry/for_hire.shtml. Accessed 16 Apr 2018
3. Taxi Info for Boro Taxis. http://www.nyc.gov/html/tlc/downloads/pdf/shl_boro_taxi_info_content_final_070313.pdf. Accessed 16 Apr 2018
4. Gelder, V.L.: Medallion Limits Stem From the 30's. *The New York Times*, 11 May 1996. <https://www.nytimes.com/1996/05/11/nyregion/medallion-limits-stem-from-the-30-s.html>. Accessed 16 Apr 2018
5. Wyman, K.: Problematic private property: the case of New York taxicab medallions. *Yale J. Regul.* **30**(169), 169 (2013)
6. Regulation and Prosperity: 1935–1960. http://www.nyc.gov/html/media/totweb/taxioftomorrow_history_regulationandprosperity.html. Accessed 16 Apr 2018
7. Wyman, K.: Problematic private property: the case of New York taxicab medallions. *Yale J. Regul.* **30**(169), 170 (2013)
8. The NYC TLC 2002 Annual Report, p. 1. http://www.nyc.gov/html/tlc/downloads/pdf/annual_report03.pdf. Accessed 16 Apr 2018
9. The Modern Taxi: 1960–2010. http://www.nyc.gov/html/media/totweb/taxioftomorrow_history_themodertaxi.html. Accessed 16 Apr 2018
10. The NYC TLC 2002 Annual Report, p. 7. http://www.nyc.gov/html/tlc/downloads/pdf/annual_report03.pdf. Accessed 16 Apr 2018
11. Schaller, B., Gilbert, G.: Villain or Bogyman? New York's Taxi Medallion System. <http://www.schallerconsult.com/taxi/taxi2.htm#first>. Accessed 16 Apr 2018
12. Wyman, K.: Problematic private property: the case of New York Taxicab medallions. *Yale J. Regul.* **30**(169), 135–139 (2013)
13. Reuters: Medallion Financial Corp. <https://www.reuters.com/finance/stocks/companyProfile/MFIN.O>. Accessed 18 Apr 2018
14. Schaller Consulting: The New York City Taxicab Factbook, p. 35. <http://www.schallerconsult.com/taxi/taxifb.pdf>. Accessed 16 Apr 2018
15. Notice of Promulgation of Rules, p. 3. http://www.nyc.gov/html/tlc/downloads/pdf/lease_cap_rules_passed.pdf. Accessed 16 Apr 2018
16. The NYC TLC 2005 Annual Report, p. 5. http://www.nyc.gov/html/tlc/downloads/pdf/2005_annual_report.pdf. Accessed 16 Apr 2018

17. Hernandez, R.: A Mysterious Hedge Fund Just Scooped Up the Foreclosed Medallions from New York City's 'Taxi King'. <http://www.businessinsider.com/nyc-taxi-king-foreclosed-medallions-scooped-up-by-hedge-fund-2017-9>. Accessed 18 Apr 2018
18. New York City Mobile Services Study. <https://www1.nyc.gov/assets/dca/MobileServicesStudy/Research-Brief.pdf>. Accessed 18 Apr 2018
19. Wyman, K.: Taxi Regulation in the Age of Uber, p. 4. <http://www.nyujlpp.org/wp-content/uploads/2017/04/Wyman-Taxi-Regulation-in-the-Age-of-Uber-20nyujlpp1.pdf>. Accessed 18 Apr 2018
20. Parkins, D.: Taming the titans. *The Economist*, 18 January 2018
21. The Economist: From zero to seventy (billion). *The Economist*, 3 September 2016
22. DMR: 90 Amazing Uber Statistics, Demographics, and Facts, March 2018. <https://expandeddrblings.com/index.php/uber-statistics/>. Accessed 18 Apr 2018
23. Wyman, K.: Problematic private property: the case of New York Taxicab medallions. *Yale J. Regul.* **30**(169), 156–164 (2013)
24. Grisworld, A.: Uber Won New York. http://www.slate.com/articles/business/moneybox/2015/11/uber_won_new_york_city_it_only_took_five_years.html. Accessed 18 Apr 2018
25. Harshbarger, R.: Yellow Cab Industry Dealt Legal Blow as It Loses Court Battle against Taxi App Companies. <https://www.amny.com/transit/nyc-yellow-cabs-lose-legal-battle-to-uber-taxi-apps-1.10825964>. Accessed 18 Apr 2018
26. Campbell, J.: NY Senate Passes Bill to Fast Track Uber, Lyft. <https://www.democratandchronicle.com/story/news/politics/albany/2017/05/17/ny-senate-passes-bill-fast-track-uber-lyft/101800922/>. Accessed 18 Apr 2018
27. Wyman, K.: Problematic private property: the case of New York taxicab medallions. *Yale J. Regul.* **30**(169), 187 (2013)



An Intelligent and Hybrid Weighted Fuzzy Time Series Model Based on Empirical Mode Decomposition for Financial Markets Forecasting

Ruixin Yang¹(✉), Junyi He³, Mingyang Xu¹, Haoqi Ni¹, Paul Jones¹,
and Nagiza Samatova^{1,2}

¹ North Carolina State University, Raleigh, NC, USA
ryang9@ncsu.edu

² Oak Ridge National Laboratory, Oak Ridge, TN, USA

³ Shanghai University, Shanghai, China

Abstract. Given the potentially high impact of accurate financial market forecasting, there has been considerable research on time series analysis for financial markets. We present a new Intelligent Hybrid Weighted Fuzzy (IHWF) time series model to improve forecasting accuracy in financial markets, which are complex nonlinear time-sensitive systems, influenced by many factors. The IHWF model uniquely combines Empirical Mode Decomposition (EMD) with a novel weighted fuzzy time series method. The model is enhanced by an Adaptive Sine-Cosine Human Learning Optimization (ASCHLO) algorithm to help find optimal parameters that further improve forecasting performance. EMD is a time series processing technique to extract the possible modes of various kinds of institutional and individual investors and traders, embedded in a given time series. Subsequently, the proposed weighted fuzzy time series method with chronological order based frequency and Neighborhood Volatility Direction (NVD) is analyzed and integrated with ASCHLO to determine the effective universe discourse, intervals and weights. In order to evaluate the performance of proposed model, we evaluate actual trading data of Taiwan Capitalization Weighted Stock Index (TAIEX) from 1990 to 2004 and the findings are compared with other well-known forecasting models. The results show that the proposed method outperforms the listing models in terms of accuracy.

Keywords: EMD · Weighted fuzzy time series
Human learning optimization algorithm · Financial markets forecasting

1 Introduction

Modeling and forecasting financial markets provide key information for risk management, portfolio construction and financial decision making. However, financial markets are notoriously complex, dynamic, nonlinear, non-stationary, nonparametric, noisy and chaotic systems [1], which can be influenced by many factors including economic cycle, government policies, and investor psychology [40].

Therefore, forecasting such systems is extremely challenging, and this area has attracted considerable worldwide attention both in research and financial communities. Although many techniques [9, 19] have been developed for forecasting financial markets, building an efficient model is still an attractive proposition since even a tiny improvement in forecasting accuracy may have a huge impact on decision making for portfolio and risk managers. Therefore, researchers continue to strive to discover ways to maximize the accuracy of financial market forecasting.

From a conventional statistical point of view, Box and Jenkins [5] established the Auto-Regressive Moving Average (ARMA) model, which combines a moving average with a linear difference equation. Although ARMA can make preliminary forecasts, it is restricted by certain requirements on the data, such as linear stationarity. To handle non-stationary datasets, the Auto-Regressive Integrated Moving Average (ARIMA) model [5], was proposed under the assumption of linearity between variables. However, conventional time series methods only deal with linear forecasting models and variables must obey a statistical normal distribution. In order to overcome this limitation, computational intelligence techniques, from a non-statistical perspective, have been applied to forecasting financial markets. However, since the native financial markets include both non-linear and vague data [24], fuzzy logic is utilized to describe daily observations and has recently been standing out as an important approach [25].

Thus, researchers are now showing an increasing interest in using Fuzzy Time Series (FTS) models for financial market forecasting [8, 23] as well as many real-world financial applications [2, 22]. The concept of FTS was first proposed by Song and Chisso to forecast university enrollment. In a financial context, Teoh et al. [26] presented a model incorporating trend-weighting into the FTS models and applied their model to the forecasting of empirical stock markets. Chen and Chang [10] handled fuzzy rules by clustering algorithms and assigned different weights to clusters and applied their model to forecast the Taiwan Stock Index (TAIEX). Zhou et al. [41] developed a portfolio optimization model combining information entropy and FTS to forecast stock exchange in Chinese financial market. Chen and Chang [10] introduced the probabilities of trends and proposed a modified fuzzy forecasting model with fuzzy-trend logical relationship. Rubio et al. [23] proposed weighted FTS model based on the weights of chronological-order and trend-order to forecast the future performance of stock market indices. In the lately financial markets with high volatility, accurate forecasting tends to be more and more difficult and challenging due to the rapid changes of trading rules and management systems.

Previous research has shown that the volatility of financial market is often influenced by different investment behaviors between various institutions and/or individual investors [38] such as investment experiences [21], short and long-term expectations [15], and even different interpretation of government policies [3]. Therefore, although the observation of the financial market is a single time series, it can be a mixture of several sub-series generated by different institutions or individuals. Thus, creating forecasting models based only on a single observed

series is not enough and may lead to bottlenecks. On the other hand, most models only depend on the historical trend indicated by the fuzzy logic group but ignore the big influence of the volatility direction indicated by neighborhood. In addition, many existing models are very sensitive to parameters settings such as the length of interval and thus forecast accuracy will be much reduced with inappropriate parameter settings in various real-world applications [12].

Therefore, to address the above weakness of existing FTS models, a novel Intelligent and Hybrid Weighted Fuzzy (IHWF) time series model is proposed in this paper. We first adopt Empirical Mode Decomposition (EMD) [16] to extract the possible modes of various kinds of institutional and individual investors embedded in an observed time series. Then, we propose a New Weighted FTS (NWF) that considers both chronological-order based frequency and the neighborhood volatility direction (NVD). The NWF will perform the proper forecasting according to various sub-series extracted by EMD and all the predicted results are aggregated as the final forecast. Finally, to optimize the NWF automatically and globally, a novel evolutionary algorithm, Adaptive Sine-Cosine Human Learning Optimization (ASCHLO), is developed and deployed. To the best of our knowledge, no existing FTS models decompose the financial time series to analyze the inner forecast process. The solution framework designed in this research is unique and significantly improves the forecast accuracy.

2 Preliminaries

This section briefly reviews the relevant studies on empirical mode decomposition, fuzzy sets and fuzzy time series.

2.1 Empirical Mode Decomposition

Empirical Mode Decomposition, introduced by Huang et al. [16] is an adaptive decomposition method for nonlinear and non-stationary time series data using

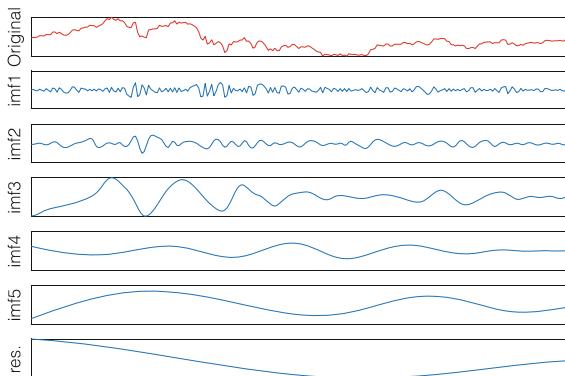


Fig. 1. The IMFs and residual of TAIEX from 2004 decomposed by EMD

the Hilbert-Huang transform (HHT). The EMD is based on the simple assumption that any signal consists of different components, thus any data may have different coexisting modes of oscillations at the same time. Given this, the time series can be decomposed into a group of zero mean and quasi-periodic signals. Each component is called an intrinsic mode function (IMF), and this is added to a residue series that represents the different scales of the original time series. Together, these form the adaptive and physical basis of the data as given in Eq. (1):

$$x(t) = \sum_{i=1}^m c_i(t) + r(t) \quad (1)$$

where $x(t)$ is the original time series data, each $c_i(t)$ represents the i th IMF, and $r(t)$ is the residual component. The detailed procedure of EMD is briefly explained in Algorithm 1.

Algorithm 1. Empirical Mode Decomposition

- 1: Identify all extrema (local maxima $x_{max}[n]$ and minima $x_{min}[n]$) of the time series $x(t)$
 - 2: Connect $x_{max}[n]$ by a cubic spline to define the upper envelope, $U(t)$, and $x_{min}[n]$ by a second cubic spline to define the lower envelope, $L(t)$
 - 3: Calculate mean of the envelope $m(t)$, i.e., $m(t) = (U(t) + L(t))/2$
 - 4: Compute the difference to obtain an IMF candidate $h(t) = x(t) - m(t)$.
 - 5: Check if $h(t)$ satisfies the condition of an IMF property, i.e., the envelopes are symmetric with respect to zero mean under certain criteria
 - 6: if $h(t)$ is an IMF, take $h(t)$ as the i th IMF, and replace $x(t)$ with the residual $R_i(t) = x(t) - h(t)$. Otherwise, repeat Steps 2 to 5 until a valid IMF function $h(t)$ is found.
-

The EMD components (IMFs and residue signal) represent different characteristics of the original signal. Figure 1, an example of the decomposed TAIEX from 2004, demonstrates this difference. As can be seen from the figure, each component has different characteristics which implies that the market is driven by different investment modes at same time. For example, IMF1 may represent High-Frequency Trading mode while residual could represent a Long-Term Investment mode.

2.2 Fuzzy Time Series

Song and Chissom [24] first introduced the concept of a fuzzy time series, based on the fuzzy set theory [37].

Definition 1 [37]. Let $U = \{u_1, u_2, \dots, u_n\}$ be the universe of discourse. The fuzzy set A in the universe of discourse U can be defined as follows:

$$A = f_A(u_1)/u_1 + f_A(u_2)/u_2 + \dots + f_A(u_n)/u_n \quad (2)$$

where f_A is the membership function of the fuzzy set A , $f_A : U \rightarrow [0, 1]$, and $f_A(u_i)$ denotes the degree of membership of u_i belonging to the fuzzy set A , with $f_A(u_i) \in [0, 1]$ and $1 \leq i \leq n$.

Definition 2 [24]. Let $Y(t)(t = \dots, 0, 1, 2, \dots)$, a subset of R (set of real numbers) be the universe of discourse in which fuzzy sets $f_i(t)(i = 1, 2, \dots)$ are defined. If $F(t)$ is a collection of $f_i(t)(i = 1, 2, \dots)$, then $F(t)$ is called a fuzzy time series on $Y(t)(t = \dots, 0, 1, 2, \dots)$.

Definition 3 [24]. Let $F(t - 1) = A_i$ and $F(t) = A_j$. The fuzzy logical relationship between $F(t - 1)$ and $F(t)$ is referred by $A_i \rightarrow A_j$, where A_i is called the left-hand side and A_j is the right-hand side of the fuzzy logical relationship.

Definition 4 [9]. Suppose there are fuzzy logical relationships with the same left-hand side A_i such that $A_i \rightarrow A_{j1}, A_i \rightarrow A_{j2}, \dots, A_i \rightarrow A_{jn}$. These fuzzy logical relationships can be grouped into a fuzzy logical relationship group, shown as $A_i \rightarrow A_{j1}, A_{j2}, \dots, A_{jn}$.

Definition 5 [23]. Let $F(t - 1) = A_i$ and $F(t) = A_j$, for $i, j = 1, 2, \dots, m$, which is denoted as $A_i \rightarrow A_j$. Then $k = j - i$ is the jump associated to this fuzzy logical relationship.

3 Adaptive Sine-Cosine Human Learning Optimization Algorithm

Evolutionary algorithms now have been successfully applied to a wide range of real-world problems of significant complexity [20, 28, 30, 32, 33]. Human Learning Optimization (HLO), which was presented recently by [27], is an evolutionary computation method inspired by the human learning mechanisms, and has been successfully used to resolve optimization problems in different applications [7, 29, 31, 35]. Due to the simple but effective structure of HLO, it does not need the gradient information in its optimization procedure, which is very suitable for the FTS optimization. In this section, we first briefly introduce the basic techniques of HLO and then propose the adaptive sine-cosine mechanism to further improve the performance of HLO.

In HLO, three learning operators - a random learning operator, an individual learning operator, and a social learning operator - are used to yield new candidates to search for the optima. These attempt to mimic human learning processes, as described in more detail in [27]. The learning operators are based on the knowledge stored in the individual knowledge data (*IKD*) and social knowledge data (*SKD*), and can be further integrated and operated to improve learning as shown in Eq. (3).

$$x_{ij} = \begin{cases} Rand(0, 1), 0 \leq rand() \leq pr \\ ik_{pj}, pr < rand() \leq pi \\ sk_{qj}, else \end{cases} \quad (3)$$

where ik and sk are individuals in IKD and SKD . pr is the probability of random learning, and the values of $(pi - pr)$ and $(1 - pi)$ represents the probabilities of performing individual learning and social learning, respectively.

Adaptive Sine-Cosine Mechanism. Based on our observation, the parameters pi and pr are extremely important. Since the search balance between the exploration and exploitation are directly determined by these two parameters, they consequently influence the performance of the algorithm. However, in basic HLO, the value of the two parameters are set with fixed value by trial and error which may have a big impact on performance, especially for various types of complex optimization problems. Therefore, we propose a new adaptive mechanism based on sine-cosine to dynamically strengthen the search efficiency and relieve the effort of the parameter setting as Eqs. (4 and 5),

$$pr = pr_{mid} + pr_{mid} \times \sin(Ite) \quad (4)$$

$$pi = pi_{mid} + pr_{mid} \times \cos(Ite + 1) \quad (5)$$

where pr_{mid} and pi_{mid} are the mid-points of fluctuation of pr and pi , respectively, and Ite is the current iteration number. In this way, various combinations of pr and pi are generated, so that ASCHLO is able to intelligently handle complex optimization cases.

4 The Intelligent and Hybrid Weighted Fuzzy Time Series Model Based on ASCHLO

We propose a novel intelligent and hybrid weighted fuzzy time series model (IHWF) based on ASCHLO. We first apply the EMD technique to decompose the original financial time series into several sub-series and then introduce a concept of the neighborhood volatility direction (NVD). By integrating two types of weights, we modify and improve the basic FTS model. These weights not only consider the chronological-order based frequency but also take NVD into consideration. Subsequently, we use this weighted FTS to develop a forecasting model for each sub-series, and to make the corresponding prediction for possible modes extracted from original observations. In this process, ASCHLO is globally utilized to optimize the associated parameters such as the partition of intervals, the weight factors both in NVD and forecast process. Finally, the final forecasting result of the original time series is calculated by aggregating all the forecasting results of extracted modes obtained through appropriate IHWF model.

4.1 The Proposed Weighted FTS Model

In our model, the trapezoidal fuzzy numbers [37] with midpoints of intervals [9] are used to analyze and derive the forecast values.

Definition 6. A fuzzy number A is said to be a trapezoidal fuzzy number, $A = (a, b, c, d)$, if its membership function has the following form:

$$\mu_{\tilde{A}}(x) = \begin{cases} 0, & x < a \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ 1, & b \leq x \leq c \\ \frac{d-x}{d-c}, & c \leq x \leq d \\ 0, & x > d \end{cases} \quad (6)$$

Definition 7. Let $A = (a_1, b_1, c_1, d_1)$ and $B = (a_2, b_2, c_2, d_2)$ be trapezoidal fuzzy numbers, and let λ be a real number. Then,

$$A \oplus B = (a_1 + a_2, b_1 + b_2, c_1 + c_2, d_1 + d_2) \quad (7)$$

$$\lambda A = \begin{cases} (\lambda a_1, \lambda b_1, \lambda c_1, \lambda d_1) & \lambda \geq 0 \\ (\lambda d_1, \lambda c_1, \lambda b_1, \lambda a_1) & \lambda \leq 0 \end{cases} \quad (8)$$

Assume there are m intervals, which are $u_1 = [d_1, d_2], u_2 = [d_2, d_3], \dots, u_{m-1} = [d_{m-1}, d_m], u_m = [d_m, d_{m+1}]$. The fuzzification process is as shown in Eq. (9) and the standard *FLRs* and *FLRGs* are determined according to Definitions 2.3 and 2.4.

$$\text{fuzzify}(F(t)) = A_i \text{ if } u(A_i) = \max[u(A_z)] \text{ for all } z \quad (9)$$

where $z = F(1), F(2), \dots, F(t)$ is the datum at time t ; and $u(A_z)$ is the degree of membership of $F(t)$ under A_z .

Following the above background on FTS, we introduce the proposed weighting rules considering the neighborhood volatility direction and chronological-order based frequency. In our model, all forecasts are based on two aspects of FTS: one is neighborhood volatility direction (*NVD*), and the other is long-term trend (*LT*) analysis.

Neighborhood Volatility Direction. Suppose we have observed stock data $F(t) = s_t$ for $t = 1, 2, \dots, N$ and they are all assigned to a fuzzified number A_i (e.g., $\text{fuzzify}(F(N)) = A_n$), let $F(t_0) \rightarrow A_p$ and $F(t_0 + 1) \rightarrow A_q$. When predicting $F(N + 1)$, traditional fuzzy logical groups (*FLRs*) only consider the historical trend based on A_n . For example, if A_n constitutes several *FLRs* in a historical time series, i.e., $A_n \rightarrow A_{k1}, A_n \rightarrow A_{k2}, \dots, A_n \rightarrow A_{kp}$, the output of the forecast will be either the mean or the median of these intervals. However, this inference has two deficiencies which may negatively impact the forecast performance: firstly, if A_n does not belong to any *FLRs* (e.g., the index reaches record highs or lows), this inference will be invalid; secondly, the model ignores short-term trends at the current point. To illustrate this, consider the time series in Fig. 2 - we now have $F(N)$ (labeled as point O) and need to forecast $F(N + 1)$. Assume there are four historical points (i.e., P_1, P_2, P_3, P_4 in Fig. 2), which have the same fuzzified number as O - traditional methods will then take all four points into consideration. However, financial markets usually have their own

inertial trend [4] and from Fig. 2, O is currently in an up-trend, indicated by its neighborhood. So it is not appropriate to take P_2 and P_4 , which could have negative impact, into account. Therefore, to address these deficiencies, we propose the concept of neighborhood volatility direction.

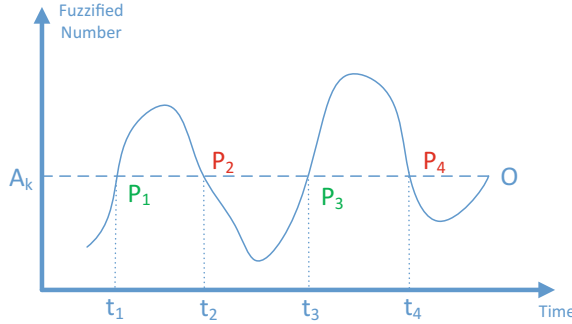


Fig. 2. Illustration of NVD

Definition 8. Let $F(t) = A_{k1}, F(t - 1) = A_{k2}, \dots, F(t - n) = A_{kn}$, where A_{kn} is fuzzified number and n is a small positive integer. The neighborhood volatility direction for $F(t)$ is denoted as $NVD(t) = [A_{k1}, A_{k2}, \dots, A_{kn}]$.

Definition 9. Suppose i, j are two data points in a certain time series, let $NVD(i) = [A_{k1}, A_{k2}, \dots, A_{kn}]$ and $NVD(j) = [A'_{k1}, A'_{k2}, \dots, A'_{kn}]$. The NVD distance (D_{NVD}) is defined as follows:

$$D_{NVD} = |A_{k1} - A'_{k1}| + \dots + |A_{kn} - A'_{kn}| \tag{10}$$

In the proposed method, the top- N historical points with the shortest distance will be determined and weighted for the current forecast by D_{NVD} . The weight associated with these points is chronological-order, as follows:

$$w_{NVD}^k = \frac{t_k}{\sum_n t} \tag{11}$$

where t_k means the time stamp of k th point selected.

Long-Term Trend. Inspired by jump theory in [23], here we also use jump metric together with chronological-order to measure the long-term trend:

Definition 10 [23]. Let $F(t - 1) = A_i$ and $F(t) = A_j$, for $i, j = 1, 2, 3, \dots$, which is denoted as $A_i \rightarrow A_j$. Then $k = j - i$ is the jump associated to this fuzzy logical relationship.

Let Δ be the jump between a *FLR*. Obviously, Δ can be negative, zero and positive (represents rise, square and fall in stock context). Similar to *NVD*, the total weight associated with Δ is calculated as:

$$w_{\Delta} = \sum_m t \quad (12)$$

where m is the number of data combinations constitute the same jump. Accordingly, the *LT* weight is calculated as:

$$w_{LT}^q = \frac{w_{\Delta}^q}{\sum_p w_{\Delta}} \quad (13)$$

where p is the number of jumps in time series.

4.2 Fuzzy Forecast Outputs

In the forecasting step, for each IMF, we propose to use a linear combination of the above weight strategies as shown in Eq. (14).

$$\tilde{A}_{t+1} = \alpha \tilde{A}_{NVD} + \beta \tilde{A}_{LT} \quad (\alpha + \beta = 1) \quad (14)$$

where

$$\tilde{A}_{NVD} = w_{NVD}^1 A_{k1} \oplus w_{NVD}^2 A_{k2} \oplus \dots \oplus w_{NVD}^o A_{kn} \quad (15)$$

$$\tilde{A}_{LT} = w_{LT}^1 A_{t+\Delta 1} \oplus w_{LT}^2 A_{t+\Delta 2} \oplus \dots \oplus w_{LT}^q A_{t+\Delta p} \quad (16)$$

and where t is the time stamp of most recently observed data. Similar as IMFs, the final forecast is calculated as Eq. (17).

$$\tilde{F}_{t+1} = \gamma \tilde{F}_{aggr} + (1 - \gamma) \tilde{F}_{LT} \quad (17)$$

where \tilde{F}_{aggr} is aggregated results of all IMFs and \tilde{F}_{LT} is calculated similar as \tilde{A}_{LT} based on actual observed data.

4.3 Algorithm for IHWF Based on ASCHLO

In this section, ASCHLO is globally utilized to find the optimal partitions of intervals in the universe of discourse and the optimal weight factors including the number of top- N points selected (*NTP*), *NVD* weights and *LT* weights simultaneously. The proposed model is now presented as follows:

- Step 1: Apply EMD to decompose the original time series $x(t)$ to IMF components $c_i(t) (i = 1, 2, 3, \dots, m)$ and one residual component $r(t)$.
- Step 2: For each IMF and residual component, let the universe of discourse U be $[D_{min} - D_1, D_{max} + D_2]$, where D_{max} and D_{min} are the maximum and the minimum values of the historical training data; D_1 and D_2 are two positive real values to allow the universe of discourse U to cover the noise of the testing data.

Step 3: Specify the control parameters of ASCHLO, such as population size, pr_{mid} , and pi_{mid} . Randomly initialize the population, including interval length l and three weights, i.e., NTP , α , β and γ for optimization. The fitness values are calculated and initial $IKDs$ and SKD are generated. Each individual performs the following sub-steps to calculate the fitness value:

- (a) Based on the generated interval length, define the fuzzy sets on U and fuzzify the historical data as Eq. (9).
- (b) Establish $FLRs$ from the fuzzy time series, and $FLRs$ and $FLRGs$ according to Definitions 2.3 and 2.4.
- (c) Defuzzify and calculate the forecast outputs. Specifically, the algorithm first calculates NVD and LT weight factors and derives the \tilde{A}_{NVD} and \tilde{A}_{LT} according to Eqs. (15) and (16), respectively. Based on the obtained \tilde{A}_{NVD} and \tilde{A}_{LT} , as well as NTP , α , β and γ generated by ASCHLO. The model can calculate \hat{A}_{t+1} according to Eq. (14). Finally, the one-step forecast will be the middle point of the interval-valued mean of \hat{A}_{t+1} .
- (d) Calculate the fitness value, i.e. the root mean square error (RMSE), as follows:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{t_{max}} (F_t - R_t)^2}{t_{max}}} \quad (18)$$

where t_{max} denotes the number of trading days of the historical training data, F_t , denotes the forecast value of the validation datum on trading day t , and R_t is the actual value of the historical validation datum on trading day t .

- Step 4: Improvise new individuals. New solutions are yielded by performing the adaptive pr and pi rule and the learning operators shown in Eqs. (4 and 5).
- Step 5: Calculate the fitness of new candidates. The fitness value of new individuals is computed as steps 3(a)–3(d).
- Step 6: Update the $IKDs$ and SKD according to the fitness of the new individuals.
- Step 7: Check the termination criterion. If the termination criterion is not met, steps 3–6 will be repeated to keep searching for the optimal parameters, otherwise go to the next step.
- Step 8: Based on the obtained optimal parameters, the model performs steps 3(a)–3(d) to calculate the final forecast on the test datasets.
- Step 9: Ultimately, the forecasting results of all extracted IMFs and residual component obtained by appropriate IHWF model are aggregated as the final forecast for the original financial time series using Eqs. (1) and (17).

Table 1. The comparison of per-year RMSE and average RMSE for different models when forecasting the TAIEX from November to December (1990 to 2004). Overall, our model (IHWF) obtains the lowest average RMSE, and is the highest ranked model.

Year	Models										IHWF
	Chen [9]	Yu [36]	Lee [18]	Huarng [17]	Egrioglu [14]	Cai [6]	Chen [11]	Cheng [13]	Wang [34]	Zhang [39]	
1990	249.31	227.15	278.94	199.65	182.42	187.10	172.89	168.77	232.58	176.25	173.12
1991	80.49	61.27	91.62	54.96	49.43	39.58	72.87	46.63	75.12	43.70	42.58
1992	60.05	67.75	78.59	61.03	50.61	39.37	43.44	45.53	64.30	40.35	39.10
1993	110.84	105.39	122.48	117.62	104.34	101.26	103.21	105.30	121.07	102.94	101.78
1994	112.07	135.29	141.13	88.34	78.50	76.32	78.63	77.17	128.96	69.38	67.83
1995	79.85	70.22	77.75	64.20	62.29	56.05	66.66	50.34	74.12	52.99	47.61
1996	54.64	54.31	59.82	52.95	51.33	49.45	59.75	53.19	55.43	51.16	52.30
1997	148.29	133.04	166.70	135.99	129.51	123.98	139.68	131.45	147.62	113.54	112.16
1998	167.84	151.27	184.55	136.46	132.07	118.41	124.44	115.89	166.94	114.93	115.05
1999	149.80	142.01	165.39	131.85	125.47	102.34	115.47	100.74	152.11	102.77	109.36
2000	176.57	191.31	277.82	121.78	156.58	131.53	123.62	125.62	225.82	150.39	129.58
2001	148.66	167.79	171.02	149.97	113.20	112.59	123.85	113.04	128.01	112.33	110.25
2002	101.40	75.11	112.45	89.13	85.95	60.33	71.98	62.94	92.07	66.41	68.85
2003	74.35	66.47	128.45	53.38	65.67	51.54	58.06	51.16	132.88	52.66	51.23
2004	82.32	72.34	75.58	67.11	61.14	50.33	57.73	54.25	58.40	54.95	54.15
Average RMSE	119.76	114.71	137.11	101.62	96.56	86.67	94.15	86.80	123.69	86.98	84.99
Ranking	9	8	10	7	6	2	5	3	11	4	1

5 Real-World Experiments and Results

5.1 Stock Market Data and Experimental Settings

To evaluate the forecast accuracy of the proposed IHWF model, the actual trading data of Taiwan Capitalization Weighted Stock Index (TAIEX) is used as benchmark. We work with daily values collected over 15 years, from January 1990 to December 2004 and also adopt the ten-month/two-month split for training/testing. That is, the historical data of the TAIEX of each year from January to October is used as the training set and the data from November to December is used as the test set. This is the same split used in existing fuzzy forecasting methods [6, 11, 13, 14, 39]. We compare the performance of the proposed IHWF model with ten well-known fuzzy time series methods using a variety of strategies [6, 9, 11, 13, 14, 17, 18, 34, 36, 39]. The performance is evaluated using RMSE as in Eq. (18).

All of the parameters for the baseline models are set to those reported in the initial publications. For our model, the population size is 30 and the NVD length is 2. pi_{mid} and pr_{mid} are set to 0.9 and 0.1 respectively. The number of iterations is fixed at 50. All implementations are in Python, and experiments were run on a PC with an Intel Core CPU i7-4790 @3.60 GHz and 16 GB RAM.

5.2 Results and Analysis

Table 1 shows a comparison of per-year RMSE and the average RMSE for different methods when forecasting the TAIEX from 1990 to 2004. We also include the

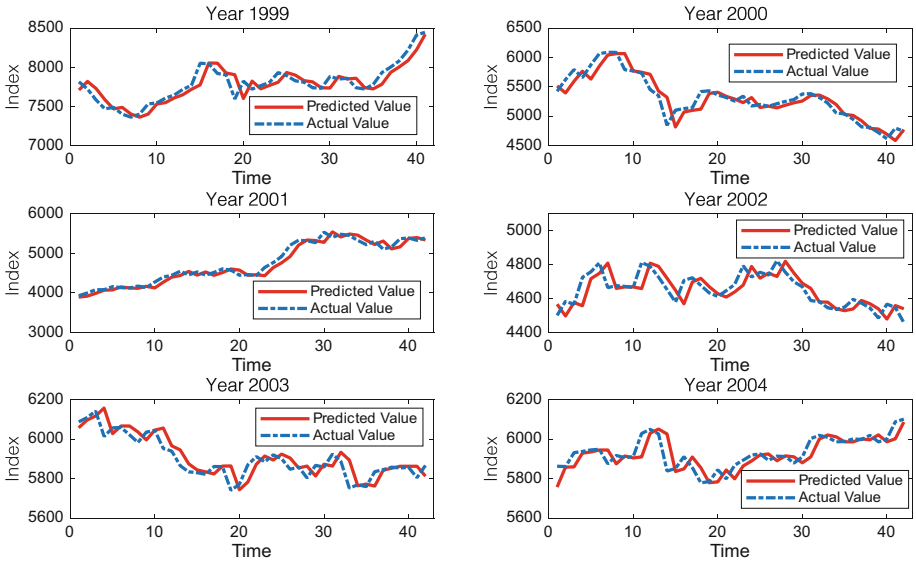


Fig. 3. Time series of daily quotes and the forecasts provided by IHWF for TAIEX from November to December (1999 to 2004). Time is measured in working days.

ranking of different methods in the bottom row. From the above comparisons, we can observe that the proposed IHWF method does not always outperform the baselines for each year, but it does best in one third of the years (5 out of 15). These kinds of mixed results are very common in the field of stock market prediction because different drivers and forces are influencing the stock market during a given time period (for instance, ‘bear’ and ‘bull’ markets). However, the long time span over which we performed our experiments does provide evidence for the efficacy of our proposed method, which leads to the lowest average RMSE shown in Table 1. Finally, Fig. 3 shows time series of daily forecasts provided by IHWF alongside the actual daily observations for the year 1999–2004. We surmise that the IHWF model is more effectively addressing the hysteresis problem of stock index forecasting, especially when the market exhibits cycles of boom (e.g., Year 2001) and bust (e.g., Year 2000). Although it is very hard to do the accurate forecast in some periods with more volatile (especially in a very short time span), our model can still capitalize the market trend which may help investors do a better decision making.

6 Conclusion

A new intelligent hybrid weighted fuzzy time series model has been presented for financial market forecasting. The model uniquely combines empirical mode decomposition with a novel weighted fuzzy time series method and is enhanced by an adaptive sine-cosine human learning parameter optimization method.

The proposed algorithm considers chronological-order based frequency and, in addition, a neighborhood volatility direction is analyzed and integrated with ASCHLO in order to determine the effective universe discourse, intervals and weights. The proposed model is evaluated against actual trading data from the TAIEX index from 1990 to 2004. Our experimental results demonstrate that the method addresses the hysteresis problem of stock market forecasting, and outperforms its counterparts over the long term, in terms of RMSE. The combination of a relatively simple implementation and effective forecasting performance suggests that our proposed model is suitable for next-generation market forecasting applications.

Acknowledgements. This material is based upon work supported in whole or in part with funding from the Laboratory for Analytic Sciences (LAS). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the LAS and/or any agency or entity of the United States Government.

References

1. Abu-Mostafa, Y.S., Atiya, A.F.: Introduction to financial forecasting. *Appl. Intell.* **6**(3), 205–213 (1996)
2. Alfonso, G., de Hierro, A.R.L., Roldán, C.: A fuzzy regression model based on finite fuzzy numbers and its application to real-world financial data. *J. Comput. Appl. Math.* **318**, 47–58 (2017)
3. Arrow, K.J., Kruz, M.: Public Investment, the Rate of Return, and Optimal Fiscal Policy, vol. 1. Routledge, Abingdon (2013)
4. Biliass, Y., Georgarakos, D., Haliassos, M.: Portfolio inertia and stock market fluctuations. *J. Money Credit Bank.* **42**(4), 715–742 (2010)
5. Box, G.E., Jenkins, G.M., Reinsel, G.C., Ljung, G.M.: *Time Series Analysis: Forecasting and Control*. Wiley, Hoboken (2015)
6. Cai, Q., Zhang, D., Zheng, W., Leung, S.C.: A new fuzzy time series forecasting model combined with ant colony optimization and auto-regression. *Knowl.-Based Syst.* **74**, 61–68 (2015)
7. Cao, J., Yan, Z., He, G.: Application of multi-objective human learning optimization method to solve AC/DC multi-objective optimal power flow problem. *Int. J. Emerg. Electr. Power Syst.* **17**(3), 327–337 (2016)
8. Chen, M.Y., Chen, B.T.: A hybrid fuzzy time series model based on granular computing for stock price forecasting. *Inf. Sci.* **294**, 227–241 (2015)
9. Chen, S.: Forecasting enrollments based on fuzzy time series. *Fuzzy Sets Syst.* **81**(3), 311–319 (1996)
10. Chen, S.M., Chang, Y.C.: Multi-variable fuzzy forecasting based on fuzzy clustering and fuzzy rule interpolation techniques. *Inf. Sci.* **180**(24), 4772–4783 (2010)
11. Chen, S.M., Chen, C.D.: TAIEX forecasting based on fuzzy time series and fuzzy variation groups. *IEEE Trans. Fuzzy Syst.* **19**(1), 1–12 (2011)
12. Chen, S.M., Phuong, B.D.H.: Fuzzy time series forecasting based on optimal partitions of intervals and optimal weighting vectors. *Knowl.-Based Syst.* **118**, 204–216 (2017)
13. Cheng, S.H., Chen, S.M., Jian, W.S.: Fuzzy time series forecasting based on fuzzy logical relationships and similarity measures. *Inf. Sci.* **327**, 272–287 (2016)

14. Egrioglu, E., Aladag, C., Yolcu, U., Uslu, V.R., Erilli, N.A.: Fuzzy time series forecasting method based on Gustafson-kessel fuzzy clustering. *Expert Syst. Appl.* **38**(8), 10355–10357 (2011)
15. Guo, X., McAleer, M., Wong, W.K., Zhu, L.: A Bayesian approach to excess volatility, short-term underreaction and long-term overreaction during financial crises. Technical report, Tinbergen Institute Discussion Paper (2016)
16. Huang, N.E., Shen, Z., Long, S.R., Wu, M.C., Shih, H., Zheng, Q., Yen, N., Tung, C.C., Liu, H.H.: The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis. In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 454, pp. 903–995. The Royal Society (1998)
17. Huarng, K., Yu, T.H.K.: Ratio-based lengths of intervals to improve fuzzy time series forecasting. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **36**(2), 328–340 (2006)
18. Lee, L.W., Wang, L.H., Chen, S.M., Leu, Y.H.: Handling forecasting problems based on two-factors high-order fuzzy time series. *IEEE Trans. Fuzzy Syst.* **14**(3), 468–477 (2006)
19. Li, C., Chiang, T.W.: Complex neurofuzzy ARIMA forecasting - a new approach using complex fuzzy sets. *IEEE Trans. Fuzzy Syst.* **21**(3), 567–584 (2013)
20. McCall, J.: Genetic algorithms for modelling and optimisation. *J. Comput. Appl. Math.* **184**(1), 205–222 (2005)
21. Menkhoff, L., Schmeling, M., Schmidt, U.: Overconfidence, experience, and professionalism: an experimental study. *J. Econ. Behav. Organ.* **86**, 92–101 (2013)
22. Ravi, K., Vad, R., Prasad, P.S.R.K.: Fuzzy formal concept analysis based opinion mining for CRM in financial services. *Appl. Soft Comput.* **60**, 786–807 (2017)
23. Rubio, A., Bermúdez, J.D., Vercher, E.: Improving stock index forecasts by using a new weighted fuzzy-trend time series method. *Expert Syst. Appl.* **76**, 12–20 (2017)
24. Song, Q., Chissom, B.S.: Forecasting enrollments with fuzzy time series-part i. *Fuzzy Sets Syst.* **54**(1), 1–9 (1993)
25. Talarposhti, F.M., Sadaei, H.J., Enayatifar, R., Guimarães, F.G., Mahmud, M., Eslami, T.: Stock market forecasting by using a hybrid model of exponential fuzzy time series. *Int. J. Approx. Reason.* **70**, 79–98 (2016)
26. Teoh, H.J., Chen, T.L., Cheng, C.H., Chu, H.H.: A hybrid multi-order fuzzy time series for forecasting stock markets. *Expert Syst. Appl.* **36**(4), 7888–7897 (2009)
27. Wang, L., Ni, H., Yang, R., Fei, M., Ye, W.: A simple human learning optimization algorithm. In: Fei, M., Peng, C., Su, Z., Song, Y., Han, Q. (eds.) *LSMS/ICSEE 2014*. CCIS, vol. 462, pp. 56–65. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45261-5_7
28. Wang, L., Ni, H., Yang, R., Pappu, V., Fenn, M.B., Pardalos, P.M.: Feature selection based on meta-heuristics for biomedicine. *Optim. Methods Softw.* **29**(4), 703–719 (2014)
29. Wang, L., Ni, H., Yang, R., Pardalos, P.M., Du, X., Fei, M.: An adaptive simplified human learning optimization algorithm. *Inf. Sci.* **320**, 126–139 (2015)
30. Wang, L., Ni, H., Yang, R., Pardalos, P.M., Jia, L., Fei, M.: Intelligent virtual reference feedback tuning and its application to heat treatment electric furnace control. *Eng. Appl. Artif. Intell.* **46**, 1–9 (2015)
31. Wang, L., Yang, R., Ni, H., Ye, W., Fei, M., Pardalos, P.M.: A human learning optimization algorithm and its application to multi-dimensional knapsack problems. *Appl. Soft Comput.* **34**, 736–743 (2015)

32. Wang, L., Yang, R., Pardalos, P.M., Qian, L., Fei, M.: An adaptive fuzzy controller based on harmony search and its application to power plant control. *Int. J. Electr. Power Energy Syst.* **53**, 272–278 (2013)
33. Wang, L., Yang, R., Xu, Y., Niu, Q., Pardalos, P.M., Fei, M.: An improved adaptive binary harmony search algorithm. *Inf. Sci.* **232**, 58–87 (2013)
34. Wang, L., Liu, X., Pedrycz, W.: Effective intervals determined by information granules to improve forecasting in fuzzy time series. *Expert Syst. Appl.* **40**(14), 5673–5679 (2013)
35. Yang, R., Xu, M., He, J., Ranshous, S., Samatova, N.F.: An intelligent weighted fuzzy time series model based on a sine-cosine adaptive human learning optimization algorithm and its application to financial markets forecasting. In: Cong, G., Peng, W.-C., Zhang, W.E., Li, C., Sun, A. (eds.) ADMA 2017. LNCS (LNAI), vol. 10604, pp. 595–607. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69179-4_42
36. Yu, H.K.: Weighted fuzzy time series models for TAIEX forecasting. *Phys. A* **349**(3), 609–624 (2005)
37. Zadeh, L.A.: Fuzzy sets. *Inf. Control* **8**(3), 338–353 (1965)
38. Zeng, X., Li, Y., Leng, S., Lin, Z., Liu, X.: Investment behavior prediction in heterogeneous information network. *Neurocomputing* **217**, 125–132 (2016)
39. Zhang, W., Zhang, S., Zhang, S., Yu, D., Huang, N.: A multi-factor and high-order stock forecast model based on type-2 FTS using cuckoo search and self-adaptive harmony search. *Neurocomputing* **240**, 13–24 (2017)
40. Zhong, X., Enke, D.: Forecasting daily stock market return using dimensionality reduction. *Expert Syst. Appl.* **67**, 126–139 (2017)
41. Zhou, R., Yang, Z., Yu, M., Ralescu, D.A.: A portfolio optimization model based on information entropy and fuzzy time series. *Fuzzy Optim. Decis. Mak.* **14**(4), 381 (2015)



Evolutionary DBN for the Customers' Sentiment Classification with Incremental Rules

Ping Yang, Dan Wang^(✉), Xiao-Lin Du, and Meng Wang

Beijing University of Technology, Beijing 100022, People's Republic of China
yangping_sx@163.com, wangdan@bjut.edu.cn

Abstract. An increasing number of reviews from the customers have been available online. Thus, sentiment classification for such reviews has attracted more and more attention from the natural language processing (NLP) community. Related literature has shown that sentiment analysis can benefit from Deep Belief Networks (DBN). However, determining the structure of the deep network and improving its performance still remains an open question. In this paper, we propose a sophisticated algorithm based on fuzzy mathematics and genetic algorithm, called evolutionary fuzzy deep belief networks with incremental rules (EFDBNI). We evaluate our proposal using empirical data sets that are dedicated for sentiment classification. The results show that EFDBNI brings out significant improvement over existing methods.

Keywords: Semi-supervised · Deep learning · Fuzzy set
Sentiment classification · Genetic algorithm

1 Introduction

Various smart appliances have play more and more important roles in our daily life, such as Apple watch, smart car etc. In particular, customers can much easier express their opinions through their smart appliances in different ways. As a result, it is not surprising that nowadays there are tons of reviews available out there. Sentiment classification for such reviews has attracted more and more attention from the natural language processing (NLP) community.

Sentiment analysis refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials. Sentiment analysis aims to determine the attitude of a speaker with respect to some topic or the overall contextual polarity of a document. such as 'positive' or 'negative', 'thumbs up' or 'thumbs down' [1].

Methods for document sentiment classification are generally based on lexicon and corpus [7, 8]. Lexicon-based approaches can derive a sentiment measure for text based on sentiment lexicons. Corpus-based approaches involve a statistical classification method. The latter usually outperforms the former and has been used in unsupervised learning [9], supervised learning and semi-supervised learning.

Early research within this field include Pang et al. [2] and Turney [3], who applied supervised learning and unsupervised learning for classifying sentiment of movie reviews, respectively. In particular, Pang et al. applied different methods based on n-gram grammar and POS (including Naïve Bayes, Maximum Entropy and SVM) to classify a review as either positive or negative. Unsupervised learning for sentiment classification works without any labeled reviews [3]. Although these methods turn out to have good performance, they all rely on labeled data which is normally difficult to obtain. Unsupervised learning of sentiment is difficult, because of the prevalence of sentimentally ambiguous reviews. SO-PMI [4] is a method for inferring the semantic orientation of a word from its statistical association with a set of positive and negative paradigm words. Further more, some scholars apply machine learning approaches to derive a classifier through supervised learning [5, 6].

Several semi-supervised learning approaches have been proposed, which use a large amount of unlabeled data together with labeled data for learning [10–12]. Sindhvani and Melville [10] propose a semi-supervised sentiment classification algorithm, which utilizes lexical prior knowledge in conjunction with unlabeled data. Recently, deep belief networks [11] is an effective model in semi-supervised learning for sentiment classification. DBN(deep belief networks) performs well in semi-supervised learning, and can be used for sentiment classification. To embedding prior knowledge in the network structure, Zhou et al. [13] propose a semi-supervised learning algorithm called fuzzy deep belief networks for sentiment classification, which is based on deep learning algorithm DBN and fuzzy sets [14].

However, there are several defects in existing semi-supervised learning methods. On one hand, they cannot deal with the data near the separating hyper-plane among classes reasonably. On the other hand, it is difficult to determine the correlation between the fuzzy sets and neurons. In this paper, we propose to enhance DBN with fuzzy mathematics and genetic algorithm in order to deal with the aforementioned challenges. In particular, our proposal maps input data to output using fuzzy rules according to their memberships of the fuzzy sets. Specifically, we design a new fuzzy set with special fuzzy rules for the data near the separating hyper-plane among the classes. And we introduced genetic algorithm to determine the correlation between the fuzzy sets and neurons. Our algorithm refers to evolutionary fuzzy deep belief networks with incremental rules (EFDBNI). We conclude that our proposal is able to tackle the aforementioned challenges and brings out better performance over existing approaches.

The remainder of this paper is organized as follows. Section 2 introduces the related work of sentiment classification. Section 3 presents our semi-supervised learning method EFDBNI in details. Section 4 presents the experimental results. We conclude the paper in Sect. 5.

2 Related Work

Many works have been proposed for sentiment classification. According to the dependence on labeled data, methods of sentiment classification fall into three categories: supervised learning, unsupervised learning and semi-supervised learning.

The study supervised learning methods for sentiment classification has begun with the work in [2]. These methods are widely used in analyzing the sentiments of various topics, such as movie reviews [15], product reviews [16, 17], microblogs [18, 19] and so on. The idea is training a domain-specific sentiment classifier for each target domain using the labeled data in that domain. Although these methods turn out to have good performance, they all rely on labeled data as training set which is normally difficult to obtain even though several works use the domain adaptation approach [20–24] as the challenge of domain-specific and annotating a large scale corpus for each domain is very expensive.

Unsupervised learning for sentiment classification is to maximize likelihood of observed data without any labeled reviews [25]. In [3], the classification of a review is predicted by the average semantic orientation of the phrases in the review that contain adjectives or adverbs. In addition, a phrase has a positive semantic orientation when it has good associations (e.g., “subtle nuances”) and a negative semantic orientation when it has bad associations (e.g., “very cavalier”). In [26], Read and Carroll investigate the effectiveness of word similarity techniques when performing weakly-supervised sentiment classification. Because labeled data are not used by unsupervised learning approaches, they are expected to be less accurate than those based on supervised learning [27].

Semi-supervised learning addresses this problem by using large amount of unlabeled data, together with the labeled data, to build better classifiers [28]. There are many semi-supervised learning methods of sentiment classification presented. To address the sentiment analysis task of rating inference, Goldberg and Zhu [29] present a graph-based semi-supervised learning algorithm which infers numerical ratings based on the perceived sentiment. In [30], a novel semi-supervised sentiment prediction algorithm utilizes lexical prior knowledge in conjunction with unlabeled examples. This method is based on joint sentiment analysis of documents and words based on a bipartite graph representation of the data. Recently, deep belief networks [11] is an effective model in semi-supervised learning for sentiment classification. DBN (deep belief networks) performs well in semi-supervised learning, and can be used for sentiment classification. To embedding prior knowledge in the network structure, Zhou et al. [13] propose a semi-supervised learning algorithm called fuzzy deep belief networks for sentiment classification. In [31], a novel sentiment analysis model is proposed based on recurrent neural network, which takes the partial document as input and then the next parts to predict the sentiment label distribution rather than the next word. In this paper, we focus on document level sentiment classification.

3 Method

We describe the procedure for training an Evolutionary Fuzzy Deep Belief Network with Incremental rules (EFDBNI). Suppose that we construct a EFDBNI with one input layer, one output layer and $N - 1$ hidden layers. Firstly, we preprocess the sentiment classification data set. Secondly, we train normal deep belief networks with one input layer, one output layer and $N - 1$ hidden layers. Thirdly, we define fuzzy sets associated with fuzzy rules. In addition, we construct special hidden layers corresponding to

these fuzzy sets based on the deep belief networks mentioned above. Then we will get a new fuzzy deep believe networks by applying membership functions to the deep structure. The whole procedure is shown in Fig. 1.

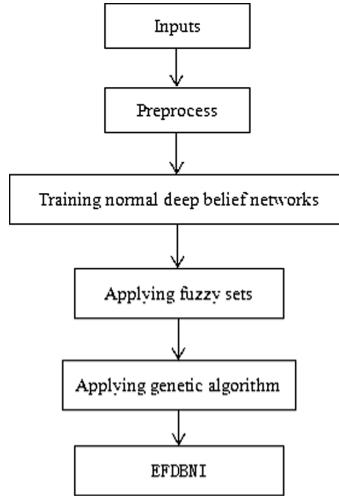


Fig. 1. The whole procedure for establishing an EFDBNI.

3.1 Preprocess

As the sentiment classification data set is normally composed of many review documents to a bag of words, we need to preprocess them in advance in the same way as that of [32].

3.2 Normal Deep Belief Networks

Preprocessed as mentioned above, each review is represented as a vector of binary weight x_i . If the j th word of the vocabulary is in the i th review, then we will set $x_j^i = 1$; otherwise, $x_j^i = 0$. Then the data set is denoted by

$$X = [x^1, x^2, \dots, x^{R+T}] \quad (1)$$

where

$$x^i = [x_1^i, x_2^i, \dots, x_D^i], i \in 1, 2, \dots, R+T \quad (2)$$

where R is the amount of training reviews, T is the amount of test reviews, D is the amount of feature words in the data set.

The L training reviews to be labeled manually is denoted by X^L . These reviews are chosen randomly. The labels corresponding to L labeled training reviews are aggregated into a set of labels Y . And it is denoted as

where

$$Y = [y^1, y^2, \dots, y^L] \quad (3)$$

$$y^i = [y_1^i, y_2^i, \dots, y_c^i]' \quad (4)$$

$$y_j^i = \begin{cases} 1, & x \in jth \text{ class} \\ 0, & x \notin jth \text{ class} \end{cases} \quad (5)$$

c is the number of classes. In sentiment classification, if a review x_i is positive, $y_i = [1, 0]'$; otherwise $y_i = [0, 1]'$.

We construct DBN with one input layer, one output layer and $N - 1$ hidden layers, while the desired EFDBNI also has one input layer, one output layer and $N - 1$ hidden layers. In both of the DBN as semi-manufacture and the EFDBNI as the made-up article, the input layer h^0 has D units and the output layer has C units. The output layer has a linear activation function. And every hidden layer uses a sigmoid function as its activation function.

We train the DBN using all reviews as inputs. Firstly, we build the DBN layer by layer using RBM through the traditional algorithm [33]. Each RBM is consisted of a binary input layer and a binary output layer [34]. Secondly, we refine the parameter space W using L labeled reviews by back-propagation. In this task, we define the optimization problem as

$$\arg \min_w f(h^N(X^L, Y^L)) \quad (6)$$

$$f(h^N(X^L), Y^L) = \frac{1}{2} \sum_{i=1}^L \sum_{j=1}^C (h_j^N(x^i) - y_j^i)^2 \quad (7)$$

where C is the number of classes and it is equal to 2 in the case of sentiment classification.

The layer h^N is obtained as follows:

$$h_t^N(x) = c_t^N + \sum_{s=1}^{D_{N-1}} w_{st}^N h_s^{N-1}(x), t = 1, 2, \dots, D_N \quad (8)$$

$$h_t^k(x) = \text{sigm}(c_t^k + \sum_{s=1}^{D_{k-1}} w_{st}^k h_s^{k-1}(x)), t = 1, 2, \dots, D; k = 1, 2, \dots, N - 1 \quad (9)$$

where w_{st}^k is the symmetric interaction term between unit s in the layer h^{k-1} and unit t in the layer h^k , $k = 1, 2, \dots, N - 1$, while c_t^k is the t th bias of layer h^k . And w_{st}^N is the symmetric interaction term between units in the layer h^{N-1} and unit t in the layer h^N . c is the t th bias of layer h^N .

3.3 Membership Function of Fuzzy Rule

Training reviews are divided into three fuzzy sets A, B and C, each of which is inferred using the fuzzy rules.

Definition 1. For a data set X , positive fuzzy set A in X is characterized by a membership function $\mu_A(x)$ which associates each review x with a real number in the interval $[0, 1]$, representing the grade of x as positive review in A :

$$A = \{(x, \mu_A(x)); x \in X\} \tag{10}$$

where

$$\mu_A(x) : X \rightarrow [0, 1] \tag{11}$$

Definition 2. For a data set X , negative fuzzy set B in X is characterized by a membership function $\mu_B(x)$ which associates each review x with a real number in the interval $[0, 1]$, representing the grade of x as negative review in B :

$$B = \{(x, \mu_B(x)); x \in X\} \tag{12}$$

where

$$\mu_B(x) : X \rightarrow [0, 1] \tag{13}$$

The membership functions are based on the value of $h^N(x)$ from the deep belief networks trained in Sect. 2.1. In the case of sentimental classification, the dimension of $h^N(x)$ is 2(corresponding to positive or negative class). Thus, the class separation line is

$$h_1^N = h_2^N \tag{14}$$

The distance between a point $h^N(x_i)$ and a separation line is

$$d(x^i) = (h_1^N(x^i) - h_2^N(x^i))/\sqrt{2} \tag{15}$$

If $d(x^i) > 0$, x^i is positive; otherwise, x^i is negative.

The two membership functions $\mu_A(x)$ and $\mu_B(x)$ are expressed as

$$\mu_A(x; \beta, \gamma) = \begin{cases} S(d(x); \gamma - \beta, \gamma - \beta/2, \gamma), & d(x) \leq \gamma \\ 1, & d(x) > \gamma \end{cases} \tag{16}$$

$$\mu_B(x; \beta, -\gamma) = \begin{cases} 1, & d(x) < -\gamma \\ 1 - S(d(x); -\gamma, -\gamma + \beta/2, -\gamma + \beta), & d(x) \geq -\gamma \end{cases} \tag{17}$$

$$S(d; \alpha, \beta, \gamma) = \begin{cases} 0, & d \leq \alpha \\ 2\left(\frac{d-\alpha}{\gamma-\alpha}\right)^2, & \alpha \leq d \leq \beta \\ 1 - 2\left(\frac{d-\gamma}{\gamma-\alpha}\right)^2, & \beta \leq d < \gamma \\ 1, & d \geq \gamma \end{cases} \tag{18}$$

If $\mu_A(x) > \mu_B(x)$ as $d(x) > 0$, the grade of membership in A is bigger than in B. If $\mu_A(x) < \mu_B(x)$ as $d(x) < 0$, the grade of membership in A is smaller than in B.

To estimate two parameters β and γ , we have

$$\gamma = \max(d(x^j)) \tag{19}$$

$$\beta = \xi \times \gamma, \xi \geq 2 \tag{20}$$

where ξ is a constant which indicates the degree of separation for the two classes.

However, it is not proper to partition the data set in this way. These fuzzy sets only model the two sentimental polarities and their fuzzy rules. Then, the input data is mapped to the output using the fuzzy rules according to their memberships of the fuzzy sets. The higher degree the input data belongs to a fuzzy set, the more proper to use its corresponding rules. In contrast, although the fuzzy sets theory allows an input data locate at the overlapping among several fuzzy sets, there are no reasonable rules can be applied to the input if it belongs to every set in a low degree. In another word, it leads to inexact results to use the existing fuzzy rules for inference with the data near the separating hyper-plane. Hence, we design a new fuzzy set with special fuzzy rules for the data near the separating hyper-plane in this paper. This new rule could compensate for the previous rules.

Definition 3. For a data set X , neutral fuzzy set C in X is characterized by a membership function $\mu_C(x)$ which associates each review x with a real number in the interval $[0, 1]$, representing the grade of x as neutral review in C :

$$C = \{(x, \mu_C(x)); x \in X\} \tag{21}$$

According to set theory, we reformulate the definition of fuzzy set C as follows:

$$C = A^c \cap B^c \tag{22}$$

where X^c represents the complementary set of a set X .

On the base of fuzzy mathematics, we have

$$\mu_C(x) = \min(1 - \mu_A(x), 1 - \mu_B(x)) \tag{23}$$

To be specific, the membership function $\mu_C(x)$ is formalized as

$$\mu_C(x) = \begin{cases} 1, & d \leq \gamma \\ 1 - 2\left[\frac{d - (\gamma - \beta)}{\beta}\right]^2, & \gamma < d \leq \gamma - \frac{\beta}{2} \\ 2\left(\frac{d - \gamma}{\beta}\right)^2, & \gamma - \frac{\beta}{2} < d \leq -\gamma + \frac{\beta}{2} \\ 1 - 2\left[\frac{d + (\gamma - \beta)}{\beta}\right]^2, & -\gamma + \frac{\beta}{2} < d \leq \gamma \\ 0, & \gamma < d \end{cases} \tag{24}$$

The parameters β and γ in (24) are as the same as the two parameters in (16) and (17). Thus, the parameters β and γ in (24) can also be estimated by (19) and (20).

3.4 Evolutionary Fuzzy Deep Belief Networks Algorithm with Incremental Rules

After extracting fuzzy parameters, deep architecture has been constructed. The architecture is shown in Fig. 2. The top hidden layer h^{N-1} is divided into three parts corresponding to each of the fuzzy rules respectively.

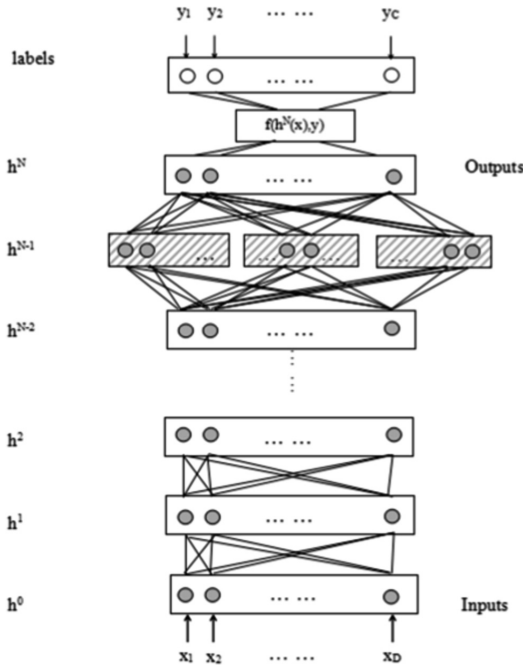


Fig. 2. Architecture of fuzzy deep belief networks.

The label of new data is denoted by \hat{j} . It is determined by

$$\hat{j} = \arg \max_j h^N(x) \tag{25}$$

The procedure of getting final outputs of the networks using $\mu_A(x)$, $\mu_B(x)$, $\mu_C(x)$ and outputs from h^{N-1} is formulated as

$$h_t^N(x) = c_t^N + \mu_A(x)P + \mu_B(x)Q + \mu_C(x)R, t = 1, 2, \dots, D_N \tag{26}$$

where

$$P = \sum_{s=1}^{D_{N-1}/3} w_{st}^N h_s^{N-1}(x) \tag{27}$$

$$Q = \sum_{s=D_{N-1}/3+1}^{2D_{N-1}/3} w_{st}^N h_s^{N-1}(x) \quad (28)$$

$$R = \sum_{s=2D_{N-1}/3+1}^{D_{N-1}} w_{st}^N h_s^{N-1}(x) \quad (29)$$

where the description of h^{N-1} can be seen in (9). However, we use genetic algorithm to determine which of units in layer h^{N-1} are used to represent each rule.

That is we determine the indexes of the hidden units in layer involve in the (27), (28) and (29). In each of the equations, the index is denoted by s . We divide the units in layer into three groups. Each of the groups is associated with one of the fuzzy rules. According to the conclusion in [35], it is helpful to determine the units associated with each fuzzy rule. As such, we need some principles to determine the features represented by the units in layer h^{N-1} associated with each fuzzy rule. Therefore, we implement this process by applying genetic algorithm (GA) for grouping problem. The fuzzy rules are groups. The features are objects to be grouped.

In our work, we only need to consider the equal group-size problem. Different from previous genetic algorithm for maximally diverse grouping problems [36, 37], our purpose is to assign the right fuzzy rules to the features represented by the units in layer h^{N-1} . Thus, we design a novel genetic algorithm which tackles this particular problem in our paper. The details are described as follows:

We divide the units in layer h^{N-1} into three manually disjoint groups. These groups are numbered with one, two and three.

As the previous genetic algorithm for grouping problems, we encode the solution as chromosome. In our work, it is suitable to use any encoding scheme which can represent the space of solutions. Thus, we adopt the most straightforward encoding scheme, namely one gene per object. For example, the chromosome 132312 would encode the solution where the first object is in group 1, the second in group 3, third in 2, fourth in 3, fifth in 1, sixth in 2.

Although there are various compositions of the groups, we can simplify all kinds of diversities among the compositions into two types. We denote three objects in three groups respectively by A, B and C. Group 1, Group 2 and Group 3 include object A, object B and object C respectively. These relationships are described in Table 1.

Table 1. Units for magnetic properties.

Group number	Index of object
1	A
2	B
3	C

Another solution to the grouping problem is to assign these three objects to different groups. We list all the cases in Table 2.

If we ignore the differences caused by capital letter (for example, the pair of ABC and BAC is equivalent to the pair of CBA and BCA), we can further simplify the cases into three. This is described in Table 3.

The differences between these solutions are induced to the assignment (grouping) of each object. So the differences can be classified into two types described in Table 3. For example, a solution is represented by chromosome 123123, while another solution is 213312. On the one hand, the diversity between the first three genes 123 of the first chromosome and the first three genes 213 of the second chromosome is the case described in the second row (the row includes the phrase ‘group number’ is the first row) and third row in Table 3. On the other hand, the diversity between the last three genes 123 of the first chromosome and the last three genes 312 of the second chromosome is the case described in the second row and fourth row in Table 3. So we can break up a pair of solution according to their consistent part and inconsistent part. Based on this fact, we design specific GA operators for our problem. The details are described as follows.

Step 1. We denote crossover rate as φ . The parameter ranges from 0.5 to 1. And it is determined manually.

Step 2. We classify the diversities between the parents into two types in Table 3. And these diversities are integrated into a set. We assign a random number each terms of the set. The number ranges from 0 to 1. If a term’s number is bigger than φ , it is added to the list for crossover.

Step 3. We make duplication of parents.

Step 4. We take the top term out of the list. And we exchange the objects in the duplication of the first parent. We only need to regroup the objects involved in the diversity term. Then the assignment of the objects are as the same as that in the second parent. At the same time, we make the assignment of the three objects in the duplication of the second parent to be as the same as the first parent. Then we repeat the process described above until the list is empty.

It should be note that the crossover rate φ is larger than 0.5. Because if it is too small, the process may create springs which are the same as parents. Since small crossover rate may lead to all of the diversities are added into the list. Then the duplication of a parent change into the chromosome which is the same as the other parent.

We redefine the mutation operator as follows:

Step 1. We denote mutation rate as ϕ . The parameter ranges from 0 to 0.5. And it is determined manually.

Step 2. Generate a random number with uniform distribution. If the number is bigger than ϕ , we will go to the next step; otherwise, quite the process.

Step 3. Pick up two groups for mutation randomly. Here, the three groups have the same probabilities to be chosen. Then mutation occurs in each of the group at a random position. The probability of a mutation of a bit in a group is equal to $3/D^{N-1}$, where D^{N-1} is the number of units in layer h^{N-1} . After we determine the positions, we exchange the group number of the two objects.

It should be note that in step 3 we keep the size of each group constantly. And we ensure that an object is only included in one group.

To evaluate the solution, we need a fitness function. Different from previous genetic algorithm for maximally diverse grouping problems, our purpose is to assign the right fuzzy rules to the features represented by the units in layer h^{N-1} . So we design a new fitness function for our problem. Further, we use the error rate of the networks that have the structure described by a solution as the criterion for fitness. To obtain the initial population, we select a number of solutions from whole possible solutions. To obtain the initial population, we select a number of solutions from whole possible solutions. It should be noted that according to the work in [35] the characteristics of a fuzzy model depend heavily on the structures rather than on the parameters of the membership functions. Further, they confirm that it is practical to select the structures before the identification of the parameters of the membership functions. In our proposed GA, we only need to determine the connections between layer h^{N-1} and h^N . In the neural networks, the structure from layer h^0 to h^{N-1} represents the membership functions. And the parameters of this structure have been well adjusted for the learning in Sect. 2.1. Thus, for one generation, we just use gradient-descent to adjust the parameters for symmetric interaction term between layer h^{N-1} and layer h^N as well as the bias of layer h^N , i.e., w^N and b^N . The process of GA is shown in Algorithm 1. The number of a generation is variable gen. The index of population is i .

Table 2. The composition of groups.

Group number	Index of object	Index of object	Index of object	Index of object	Index of object	Index of object
1	A	B	A	C	C	B
2	B	A	C	B	A	C
3	C	C	B	A	B	A

Table 3. The composition of groups.

Group number	Index of object	Index of object	Index of object
1	A	B	C
2	B	A	A
3	C	C	B

Algorithm 1. Genetic Algorithm.

```

Initialization: initial population of  $\tau$  individuals, set max generation as maxGen (an
integer), selection rate  $\omega$ ;
for gen in 1 to maxGen
  for individual i in population
    Train the neural networks, whose structure is determined by the solution
    represented by individual i;
    Calculate the fitness value of individual i by error rate;
  end
  Resort the individuals in descending order by fitness value;
  Select top  $\omega \times p$  individuals as parents into a set p.
  j=1;
  while j < 0.5 *  $\omega \times p$ 
    Select a pair of parent individuals randomly from set p;
    Implement crossover and mutation to the parents and create two
    off-springs;
    Replace two worst individuals in the current population by the off-springs.
    j++;
  end
end

```

After we determine the structure of the whole networks using GA, we will identify all parameters of the whole fuzzy deep belief networks. This is supervised learning. The EFDBNI algorithm also uses gradient-descent to retrain the parameters through the whole deep architecture. The optimization problem is the same as the one described in (6) and (7). Hence, the whole fuzzy deep belief networks algorithm is shown in Algorithm 2.

Algorithm 2. Algorithm of EFDBNI.

```

Input: data  $X, Y^L$ ;
number of units in every hidden layer  $D_1, D_2, \dots, D_N$ ;
number of layers  $N$ ; number of epochs  $Q$ ;
number of training data  $R$ ; number of test data  $T$ ;
hidden layer  $h^1, h^2, \dots, h^{N-1}$ ;
parameter space  $W = \{w^1, w^2, \dots, w^N\}$ ; biases  $b, c$ ;
Output: deep architecture with parameter space  $W$ 
1. Estimated parameters of  $\mu_A(x), \mu_B(x)$  and  $\mu_C(x)$ 
(1) Greedy layer-wise unsupervised learning using Eq.(8) and Eq.(9).
(2) Supervised learning with DBN architecture using Eq. (6) and (7).
(3) Estimate parameters based on Eqs. (19) and(20).
2. Refine the EFDBNI using  $X^L, Y^L, \mu_A(x), \mu_B(x)$  and  $\mu_C(x)$ 
(1) Use GA to determine the architecture of the EFDBNI.
(2) Supervised learning with EFDBNI architecture using Eq. (6) and (7).
(3) Classify the reviews based on the trained FDBN architecture using Eq. (25).

```

4 Results and Discussion

4.1 Experimental Setup

We use three sentiment classification data sets. The data sets includes electronics (ELE), restaurants (RES) and movies (MOV). Each of them contains 1000 positive and 1000 negative reviews.

We divide the 2000 reviews into two parts. Half of the reviews are randomly selected as training data and the remaining reviews are used for testing. Only 10% of the training reviews are labeled.

We set all of the neural networks consist of one input layer, one output layer and three hidden layers. And there are 100, 100, and 200 hidden units in the three hidden layers respectively. However, the structures of the neural networks are different among the three data sets. Because the number of units in the input layer is the same as the dimensions of each data set. The max number of iterations of unsupervised-learning is set to 1000, and the supervised-learning is repeat for 10 times for each labeled data.

The parameter ξ is set by experience for different data sets. When $\xi = 3$, EFDBNI can get relatively better results on all data sets. Thus, we set ξ as 3. Here, we set the max generation of GA as 100.

4.2 Performance Comparison

We compare the classification performance of EFDBNI with two representative semi-supervised learning classifiers, i.e., transductive SVM (TSVM) [38] and Fuzzy deep belief networks (FDBN) [13].

The test accuracy on three data sets with three rules can be seen in Fig. 3, we can see that the performance of the proposed method is better than the others on all three data sets.

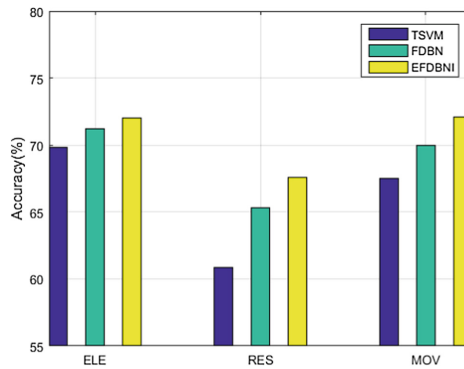


Fig. 3. Test accuracy with 100 labeled reviews on three data sets for TSVM, FDBN and EFDBNI.

5 Conclusion

This paper proposes a novel semi-supervised learning algorithm EFDBNI to address the sentiment classification problem with a small number of labeled reviews. Our proposal inherits the advantages of previous works about deep learning for sentimental classification, and has significantly improved the performance of existing deep learning architecture.

Acknowledgement. This work was supported by Beijing Natural Science Foundation P.R. China (4173072).

References

1. Li, S., Lee, S.Y.M., Chen, Y., Huang, C., and Zhou, G.: Sentiment classification and polarity shifting. In: Proceedings of the 23rd International Conference on Computational Linguistics, pp. 635–643 (2010)
2. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? Sentiment classification using machine learning techniques. In: Proceedings of EMNLP-2002, pp. 79–86 (2002)
3. Turney, P.: Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In: Annual Meeting of the Association of Computational Linguistics, pp. 417–424 (2002)
4. Turney, P., Littman, M.: Measuring praise and criticism: inference of semantic orientation from association. *ACM Trans. Inf. Syst.* **21**, 315–346 (2003)
5. McDonald, R., Hannan, K., Neylon, T., Wells, M., Reynar, J.: Structured models for fine-to-coarse sentiment analysis. In: Annual Meeting of the Association of Computational Linguistics, pp. 432–439 (2007)
6. Xia, Y., Wang, L., Wong, K.-F., Xu, M.: Lyric-based song sentiment classification with sentiment vector space model. In: Annual Meeting of the Association of Computational Linguistics, pp. 133–136 (2008)
7. Wan, X.: Co-training for cross-lingual sentiment classification. In: IEEE Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, pp. 235–243 (2009)
8. Pang, B., Lee, L.: A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In: Annual Meeting of the Association of Computational Linguistics, pp. 271–278. Association for Computational Linguistics, Barcelona (2004)
9. Zagibalov, T., Carroll, J.: Automatic seed word selection for unsupervised sentiment classification of Chinese text. In: International Conference on Computational Linguistics, pp. 1073–1080 (2008)
10. Sindhvani, V., Melville, P.: Document-word co-regularization for semi-supervised sentiment analysis. In: IEEE International Conference on Data Mining, pp. 1025–1030 (2008)
11. Hinton, G.E., Osindero, S., Teh, Y.-W.: A fast learning algorithm for deep belief nets. *Neural Comput.* **18**, 1527–1554 (2006)
12. Zhou, S., Chen, Q., Wang, X.: Active deep networks for semi-supervised sentiment classification. In: International Conference on Computational Linguistics, Poster, pp. 1515–1523 (2010)
13. Zhou, S., Chen, Q., Wang, X.: Fuzzy deep belief networks for semi-supervised sentiment classification. *Neurocomputing* **131**, 312–322 (2014)
14. Zadeh, A.: Fuzzy sets. *Inf. Control* **8**, 338–353 (1965)
15. Zhuang, L., Jing, F., Zhu, X.-Y.: Movie review mining and summarization. In: Proceedings of the 15th ACM International Conference on Information and Knowledge Management, pp. 43–50. ACM (2006)
16. Hu, M., Liu, B.: Mining and summarizing customer reviews. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 168–177 (2004)

17. Dave, K., Lawrence, S., Pennock, D.M.: Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In: Proceedings of the 12th International Conference on World Wide Web, pp. 519–528. ACM (2003)
18. Go, A., Bhayani, R., Huang, L.: Twitter Sentiment Classification Using Distant Supervision. CS224N Project Report, pp. 1–12. Stanford (2009)
19. Wu, F., Song, Y., Huang, Y.: Microblog sentiment classification with contextual knowledge regularization. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, pp. 2332–2338 (2015)
20. Aue, A., Gamon, M.: Customizing sentiment classifiers to new domains: a case study. In: International Conference on Recent Advances in Natural Language Processing (2005)
21. Tan, S., Wu, G., Tang, H., Cheng, X.: A novel scheme for domain-transfer problem in the context of sentiment analysis, pp. 979–982 (2007)
22. Blitzer, J., Dredze, M., Pereira, F.: Biographies, bollywood, boom-boxes and blenders: domain adaptation for sentiment classification. In: Annual Meeting of the Association of Computational Linguistics, pp. 440–447 (2007)
23. Li, S., Zong, C.: Multi-domain sentiment classification. In: Annual Meeting of the Association of Computational Linguistics, pp. 257–260. Association for Computational Linguistics (2008)
24. Pan, S.J., Ni, X., Sun, J., Yang, Q., Chen, Z.: Cross-domain sentiment classification via spectral feature alignment. In: International World Wide Web Conference, pp. 751–760. ACM (2010)
25. Li, S., Huang, C., Zhou, G., Lee, S.Y.M.: Employing personal/impersonal views in supervised and semi-supervised sentiment classification. In: Annual Meeting of the Association for Computational Linguistics, pp. 414–423. Association for Computational Linguistics, Uppsala (2010)
26. Read, J., Carroll, J.: Weakly supervised techniques for domain-independent sentiment classification. In: Proceedings of the 1st International CIKM Workshop on Topic-Sentiment Analysis for Mass Opinion, TSA 2009, pp. 45–52. ACM, New York (2009)
27. Silva, D.N., Coletta, L., Hruschka, E., Hruschka, E.J.: Using unsupervised information to improve semi-supervised tweet sentiment classification. *Inf. Sci.* **355–356**, 348–365 (2016)
28. Zhu, X.: Semi-supervised learning literature survey. Ph.D. thesis (2007)
29. Goldberg, A.B., Zhu, X.: Seeing stars when there aren't many stars: graph-based semi-supervised learning for sentiment categorization. In: Proceedings of TextGraphs: The First Workshop on Graph Based Methods for Natural Language Processing, pp. 45–52. Association for Computational Linguistics (2006)
30. Sindhwani, V., Melville, P.: Document-word co-regularization for semi-supervised sentiment analysis. In: International Conference on Data Mining, pp. 1025–1030. IEEE, Pisa (2008)
31. Rong, W., Peng, B., Ouyang, Y., Li, C., Xiong, Z.: Structural information aware deep semi-supervised recurrent neural network for sentiment analysis. *Front. Comput. Sci.* **9**(2), 171–184 (2015)
32. Dasgupta, S., Ng, V.: Mine the easy, classify the hard: a semi-supervised approach to automatic sentiment classification. In: Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, pp. 701–709 (2009)
33. Bengio, Y.: Learning deep architecture for AI. *Found. Trends Mach. Learn.* **2**, 1–127 (2009)
34. Smolensky, P.: Information processing in dynamical systems: foundations of harmony theory. In: *Parallel Distributed Processing: Explorations in the Micro structure of Cognition*, vol. 1, pp. 194–281 (1986)

35. Lin, C.T., Lee, C.S.G.: Neural-network-based fuzzy logic control and decision system. *IEEE Trans. Comput.* **40**, 1320–1336 (1991)
36. Falkenauer, E.: A genetic algorithm for grouping. In: *Proceedings of the Fifth International Symposium on Applied Stochastic Models and Data Analysis*, pp. 198–206 (1991)
37. Smith, D.: Bin packing with adaptive search. In: *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, pp. 202–207 (1985)
38. Kamvar, S., Klein, D., Manning, C.: Spectral learning. In: *International Joint Conferences on Artificial Intelligence*, pp. 561–566. AAAI Press, Catalonia (2003)



Clustering Professional Baseball Players with SOM and Deciding Team Reinforcement Strategy with AHP

Kazuhiro Kohara^(✉) and Shota Enomoto

Chiba Institute of Technology, 2-17-1 Tsudanuma,
Narashino, Chiba 275-0016, Japan
kohara.kazuhiro@it-chiba.ac.jp

Abstract. In this paper, we propose an integration method that uses self-organizing maps (SOM) and the analytic hierarchy process (AHP) to cluster professional baseball players and to make decision on team reinforcement strategy. We used data of pitchers in the Japanese professional baseball teams. First, we collected data of 302 pitchers and clustered these pitchers using the following fourteen features: number of games pitched, number of wins, number of loses, number of save, number of hold, number of innings pitched, rate of strikeout, ERA (earned run average), percentage of hits a pitcher allows, WHIP (walks plus hits per inning pitched), K/BB (strikeout to walk ratio), FIP (fielding independent pitching), LOB% (left on base percentage), RSAA (runs saved above average). Second, we created pitcher maps of all teams and each team with SOM. Third, we examined main features of each cluster. Fourth, we considered team reinforcement strategies by using the pitcher maps. Finally, we used AHP to determine the team reinforcement strategy.

Keywords: Clustering · Visualization · Data mining
Business intelligence · Sport industry · Baseball · Decision making
Self-organizing maps · AHP

1 Introduction

Machine learning and data mining techniques have been extensively investigated, and various attempts have been made to apply them to baseball e.g., [1–5]. Tolbert and Trafalis applied SVM (Support Vector Machine) to predicting MLB (Major League Baseball) championship winners [1]. Ishii applied K-means clustering to identifying undervalued baseball players [2]. Pane applied K-means clustering and Fisher-wise criterion to identifying clusters of MLB pitchers [3]. Tung applied PCA (Principal Component Analysis) and K-means clustering to analyzing a multivariate data set of career batting performances in MLB [4]. Vazquez applied time series and clustering algorithms to predicting baseball results [5]. In this paper, we propose an integration method that uses Self-Organizing Maps (SOM) [6] and the analytic hierarchy process (AHP) [7] to cluster professional baseball players and to make decision on team reinforcement strategy. We used data of pitchers in Japanese baseball teams. First, we collected data of 302 pitchers and clustered these pitchers using fourteen features. Second,

we created pitcher maps of all teams and each team with SOM. Third, we examined main features of each cluster. Fourth, we considered team reinforcement strategies by using pitcher maps. Finally, we used AHP to determine the team reinforcement strategy.

2 Clustering Professional Baseball Players with SOM

The SOM algorithm is based on unsupervised, competitive learning [6]. It provides a topology preserving mapping from the high dimensional space to map units. Map units, or neurons, usually form a two-dimensional lattice and thus the mapping is a mapping from high dimensional space onto a plane.

Previously, we proposed a way of purchase decision support using SOM and AHP. First, we provided two class boundaries, which divide the range between the maximum and minimum of an input feature value into three equal parts. Second, we created self-organizing product maps using the classified inputs. We applied our way to five kinds of products and confirmed its effectiveness [8]. When we previously compared SOM with the other clustering algorithms (hierarchical clustering and K-means clustering) for product clustering, SOM were superior to the other clustering algorithms for both visibility and clustering ability [9]. Therefore, we used SOM for baseball players clustering.

We used data of pitchers of NPB (Nippon Professional Baseball Organization) [10]. We collected data of 302 pitchers in 2015 from Japanese professional baseball database [10, 11]. We clustered these pitchers using the following fourteen features: number of games pitched, number of wins, number of loses, number of save, number of hold, number of innings pitched, rate of strikeouts, ERA (earned run average), percentage of hits a pitcher allows, WHIP (walks plus hits per inning pitched), K/BB (strikeout to walk ratio), FIP (fielding independent pitching), LOB% (left on base percentage), RSAA (runs saved above average).

In each feature, we provide two class boundaries, which divide the range between the maximum and minimum of an input feature value into three equal parts. For classifying the data of the number of games pitched, we divided the number into three classes: under 27, over 28 to 50, and over 51. For classifying the data of the number of wins, we divided the number into three classes: under 5, over 6 to 10, and over 11. For classifying the data of the number of loses, we divided the number into three classes: under 4, over 5 to 8, and over 9. For classifying the data of the number of save, we divided the number into three classes: under 13, over 14 to 27, and over 28. For classifying the data of the number of hold, we divided the number into three classes: under 13, over 14 to 26, and over 27. For classifying the data of the number of innings pitched, we divided the number into three classes: under 74, over 75 to 140, and over 141. For classifying the data of the rate of strikeouts, we divided the rate into three classes: under 6.09, over 6.10 to 10.15, and over 10.16. For classifying the data of ERA, we divided ERA into three classes: under 3.52, over 3.53 to 6.64, and over 6.65. For classifying the data of the percentage of hits a pitcher allows, we divided the percentage into three classes: under 8.35, over 8.36 to 13.08, and over 13.09. For classifying the data of WHIP, we divided WHIP into three classes: under 1.36, over 1.37 to 2.08, and over 2.09. For classifying the data of K/BB, we divided K/BB into three classes: under 4.70, over 4.71 to 8.85, and over 8.86. For classifying the data of FIP, we divided FIP into three classes: under 3.20, over 3.21 to 5.27, and over 5.28.

For classifying the data of LOB%, we divided LOB% into three classes: under 0.661, over 0.662 to 0.814, and over 0.815. For classifying the data of RSAA, we divided RSAA into three classes: under -2.1083, over -2.1082 to 16.65, and over 16.66.

Table 1 shows a part of the feature matrix for pitchers.

Table 1. A part of the feature matrix for pitchers.

Name	Number of games pitched			Number of wins		
	Under 27	Over 28 to 50	Over 51	Under 5	Over 6 to 10	Over 11
Makita	0	1	0	0	1	0
Hamada	1	0	0	1	0	0
Sawamura	0	0	1	0	1	0
Settu	1	0	0	0	0	1
Wakui	0	1	0	0	0	1
Arihara	1	0	0	0	1	0
Masui	0	0	1	1	0	0
Fujinami	0	1	0	0	0	1
Yamaguchi	0	0	1	1	0	0

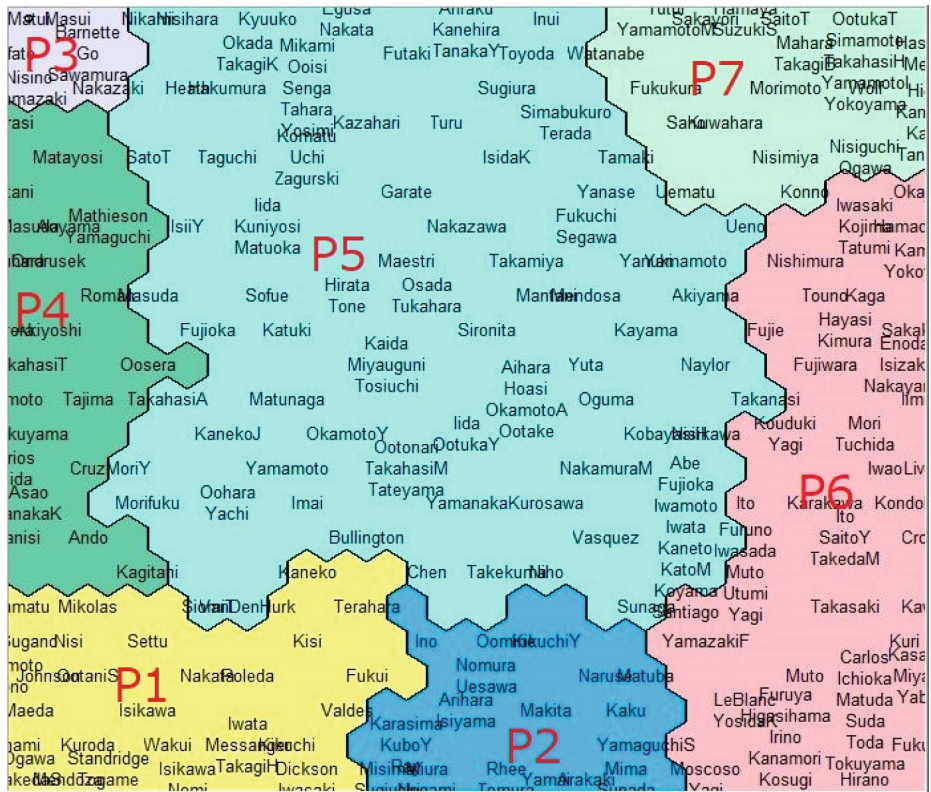


Fig. 1. Self-organizing cluster map of pitchers of all teams.

We inputted the data of all pitchers into SOM and created pitcher maps of all teams. Figure 1 shows self-organizing map of pitchers of all teams. Figures 2, 3 and 4 show examples of component maps of pitchers.

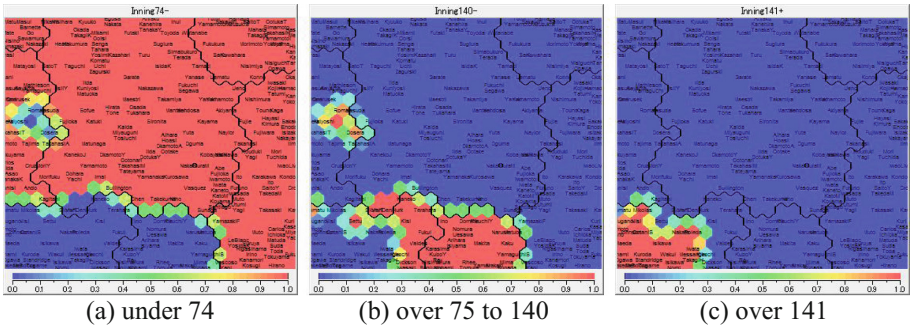


Fig. 2. Component map of pitchers of all teams: number of innings pitched. (Color figure online)

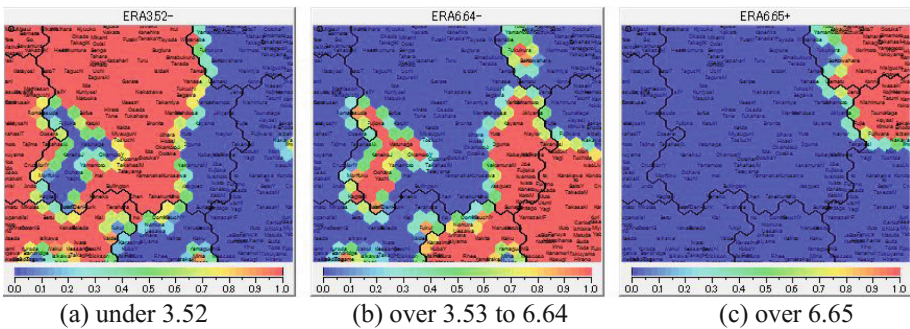


Fig. 3. Component map of pitchers of all teams: ERA (Earned Run Average). (Color figure online)

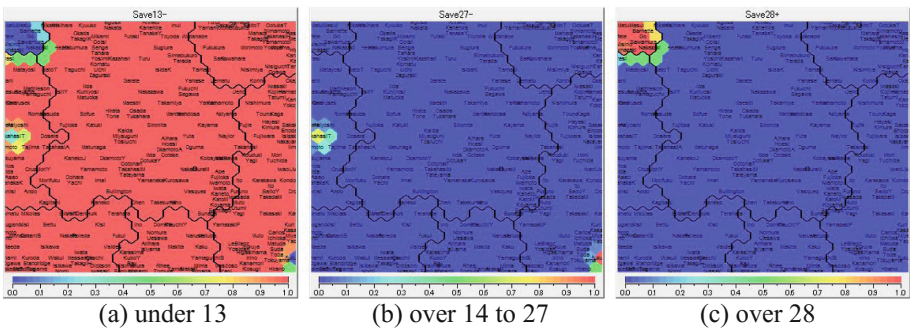


Fig. 4. Component map of pitchers of all teams: number of save. (Color figure online)

There were seven clusters in Fig. 1. When inspecting component maps, the feature of each cluster is clear. For example, red neurons correspond to over 141 innings pitched in Fig. 2(c) and red neurons correspond to over 75 to 140 innings pitched in Fig. 2(b). Red neurons correspond to under 3.52 ERA in Fig. 3(a) and red neurons correspond to over 3.53 to 6.64 ERA in Fig. 3(b). Red neurons correspond to over 28 save in Fig. 4(c).

As the number of innings pitched is large (over 141) and ERA is small (under 3.52) in cluster P1, a pitcher belonging to P1 is one of the best starting pitcher. As the number of innings pitched is medium (over 75 to 140) and ERA is medium (over 3.53 to 6.64) in cluster P2, a pitcher belonging to P2 is one of the second best starting pitcher. As the number of save is large (over 28) and ERA is small (under 3.52) in cluster P3, a pitcher belonging to P3 is a *closer*. We inspected every component maps and understand that features of Clusters P1 to P7 are as shown in Table 2.

Table 2. Main features of all NPB pitchers in 2015 in each cluster.

Cluster	Features	Main feature
P1	Number of innings pitched is large Both ERA and WHIP are small	Best starting pitchers
P2	Number of innings pitched is medium Both ERA and WHIP are medium	Second best starting pitchers
P3	Number of save is large	Closer
P4	Number of hold is large	Best setup pitchers
P5	Number of wins and loses is small	Third best starting pitchers or second best setup pitchers
P6	Number of wins and loses is small Both ERA and WHIP are large	Fourth best starting pitchers or third best setup pitchers
P7	Number of innings pitched is small Both ERA and WHIP are large	Bad pitchers

3 Considering Team Reinforcement Strategies

Next, we inputted the data of pitchers belonging to Chiba Lotte Marines into SOM and created pitcher maps. Figure 5 shows self-organizing pitcher maps of Lotte.

We inspected every component maps and understand that main feature of Clusters L1 to L6 are as shown in Table 3.

Here, we assumed that organization of pitchers in a strong team is as follows: the number of starting pitchers is five to six, the number of setup pitchers is one to two, the number of closer is one to two, and the number of relief pitchers is three to five.

When comparing Lotte's organization of pitchers with a strong team's organization, we understand that the number of starting pitchers is not enough.

Here, we chose alternatives for reinforcement strategies of starting pitchers as follows.

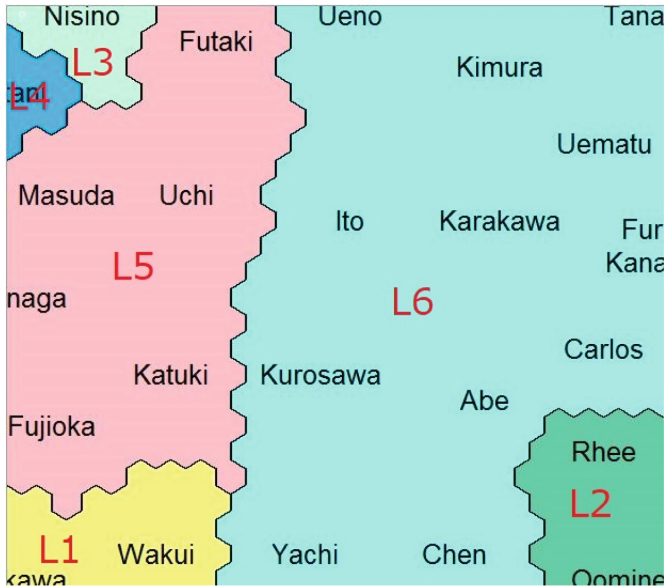


Fig. 5. Self-organizing cluster map of pitchers of Chiba Lotte Marines.

Table 3. Main features of pitchers of Chiba Lotte Marines in 2015 in each cluster.

Cluster	Main feature (# of pitchers)	Name (cluster in all NPB pitchers)
L1	Best starting pitchers (2)	Wakui, Isikawa (P1)
L2	Second best starting pitchers (2)	Rhee, Oomine (P2)
L3	Closer (1)	Nisino (P3)
L4	Best setup pitchers (1)	Ootani (P4)
L5	Second best setup pitchers (6)	Masuda, Fujioka, Matunaga, Uchi, Niki (P5), katuki (P6)
L6	Substitutes (13)	Kurosawa, Yachi, Abe, Chen (P5), Kanamori, Furuya, Carlos (P6)

Step 1: We choose pitchers (1) who belong to Clusters P1, P2, P5 or P6, (2) whose contract have been expired or who declared *free agent*, and (3) whose number of innings pitched was large or whose percentage of hits he allows was small. We chose Stanridge and Bullington.

Step 2: We choose pitchers (1) who belong to Clusters P1, P2, P5 or P6, (2) who are young and whose salary is low, (3) whose number of innings pitched was medium or whose FIP was small or whose RSAA was not small. We chose Iida and Mima.

Table 4 shows the data of four alternatives for a reinforced starting pitcher.

Table 4. Data of alternatives for a reinforced starting pitcher.

Name	Innings pitched	Hit ratio	RSAA	FIP	Salary (million yen)	Age	Right/left
Standridge	144.3	9.4	-0.52	3.79	200	37	right
Bullington	73.6	7.3	2.378	3.18	150	35	right
Mima	86.3	10.6	-7.035	3.53	40	29	right
Iida	41.3	6.5	2.169	3.19	4	24	left

Hit ratio: percentage of hits a pitcher allows,

Right/left: right throw or left throw.

4 Decision Making on Team Reinforcement Strategy with AHP

AHP is a multi-criteria decision method that uses hierarchical structures to represent a problem [7]. Pairwise comparisons are based on forming a judgment between two particular elements rather than attempting to prioritize an entire list of elements. The AHP scales of pairwise comparisons are shown in Table 5.

Table 5. The AHP scales for pairwise comparisons.

Intensity of importance	Definition and explanation
1	Equal importance
3	Moderate importance
5	Essential or strong importance
7	Demonstrated importance
9	Extreme importance
2, 4, 6, 8	Intermediate values between the two adjacent judgments when compromise is needed

Figure 6 shows an example of the relative measurement AHP model created for the task of deciding a high capable pitcher. Here, we used the following four criteria: innings pitched, hit ratio (percentage of hits a pitcher allows), RSAA and FIP.

We assumed the pairwise comparison matrix for Ciba Lotte Marines. The pairwise comparison matrix for the four criteria is shown in Table 6. Here, we assumed that large innings pitched is most important, small hit ratio is second most important, and small FIP is third most important. As a result, innings pitched is most important and its weight is 0.565.

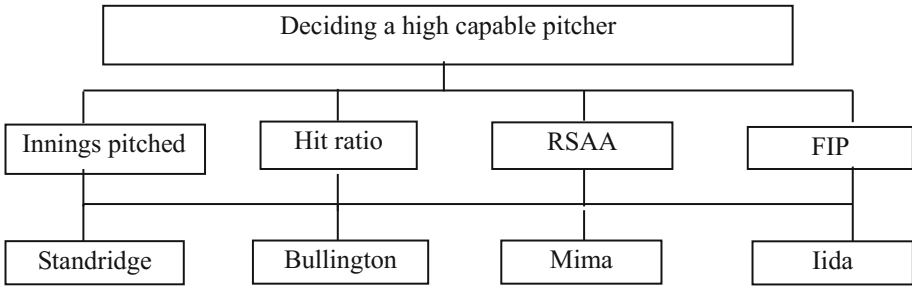


Fig. 6. AHP model for deciding a high capable pitcher.

Table 6. Pairwise comparisons of four criteria.

	Innings pitched	Hit ratio	RSAA	FIP	Weight
Innings pitched	1	3	7	5	0.565
Hit ratio	1/3	1	5	3	0.262
RSAA	1/7	1/5	1	1/3	0.055
FIP	1/5	1/3	3	1	0.118

Consistency index = 0.039

Consistency index shows whether the pairwise comparison is appropriate or not. When the index is lower than 0.1, the pairwise comparison is appropriate. When the index is over 0.1, the comparison is not appropriate and should be corrected. In this case, consistency index was 0.01 and the pairwise comparison was appropriate.

The pairwise comparisons of four alternatives with respect to innings pitched are shown in Table 7. The weight of Standridge was highest, because the number of innings pitched of Standridge was largest.

Table 7. Pairwise comparisons of alternatives with respect to innings pitched.

	Standridge	Bullington	Mima	Iida	Weight
Standridge	1	6	5	8	0.636
Bullington	1/6	1	1/2	5	0.127
Mima	1/5	2	1	6	0.195
Iida	1/8	1/5	1/6	1	0.042

Consistency index = 0.086

The pairwise comparisons of four alternatives with respect to hit ratio are shown in Table 8. The weight of Iida was highest, because the hit ratio of Iida was smallest.

Table 8. Pairwise comparisons of alternatives with respect to hit ratio.

	Standridge	Bullington	Mima	Iida	Weight
Standridge	1	1/2	2	1/3	0.154
Bullington	2	1	4	1/2	0.288
Mima	1/2	1/4	1	1/5	0.081
Iida	3	2	5	1	0.477

Consistency index = 0.007

The pairwise comparisons of four alternatives with respect to RSAA are shown in Table 9. The weight of Bullington was highest, because the RSAA of Bullington was largest.

Table 9. Pairwise comparisons of alternatives with respect to RSAA.

	Standridge	Bullington	Mima	Iida	Weight
Standridge	1	1/5	2	1/2	0.125
Bullington	5	1	6	3	0.577
Mima	1/2	1/6	1	1/3	0.077
Iida	2	1/3	3	1	0.222

Consistency index = 0.011

The pairwise comparisons of four alternatives with respect to FIP are shown in Table 10. The weight of Bullington was highest, because the FIP of Bullington was smallest.

Table 10. Pairwise comparisons of alternatives with respect to FIP.

	Standridge	Bullington	Mima	Iida	Weight
Standridge	1	1/6	1/2	1/3	0.079
Bullington	6	1	5	2	0.533
Mima	2	1/5	1	1/2	0.130
Iida	3	1/2	2	1	0.253

Consistency index = 0.008

Table 11 shows final results of AHP. Standridge was the most capable pitcher, because we assumed that large innings pitched is most important and small hit ratio is second most important. The number innings pitched of Standridge is largest.

Table 11. Final results of deciding a high capable pitcher.

Criteria	Innings pitched	Hit ratio	RSAA	FIP	Result
Weight of criteria	0.565	0.262	0.055	0.118	
Standridge	0.636	0.154	0.125	0.079	0.416
Bullington	0.127	0.288	0.577	0.533	0.242
Mima	0.195	0.081	0.077	0.130	0.151
Iida	0.042	0.477	0.222	0.253	0.191

Figure 7 shows an example of the relative measurement AHP model created for the task of deciding a reinforced starting pitcher. Here, we used the following five criteria: capability, salary, age, right/left throw and feasibility.

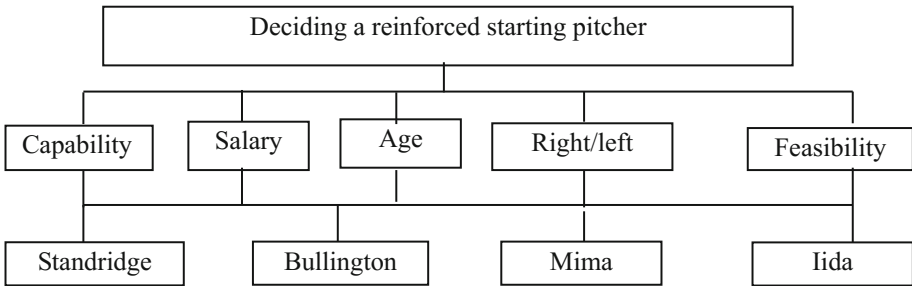


Fig. 7. AHP model for deciding a reinforced starting pitcher.

We assumed the pairwise comparison matrix for Chiba Lotte Marines. The pairwise comparison matrix for the five criteria is shown in Table 12. Here, we assumed that capability and feasibility are most important, and right/left throw is third most important. As a result, capability and feasibility are most important and their weights are 0.362.

Table 12. Pairwise comparisons of five criteria.

	Capability	Salary	Age	Right/left	Feasibility	Weight
Capability	1	7	5	3	1	0.362
Salary	1/7	1	1/3	1/5	1/7	0.039
Age	1/5	3	1	1/3	1/5	0.076
Right/left	1/3	5	3	1	1/3	0.161
Feasibility	1	7	5	3	1	0.362

Consistency index = 0.034

The weights of four alternatives with respect to capability are shown in Table 11.

The pairwise comparisons of four alternatives with respect to salary are shown in Table 13. The weight of Iida was highest, because the salary of Iida was cheapest.

Table 13. Pairwise comparisons of alternatives with respect to salary.

	Standridge	Bullington	Mima	Iida	Weight
Standridge	1	1/2	1/4	1/6	0.074
Bullington	2	1	1/2	1/4	0.138
Mima	4	2	1	1/2	0.275
Iida	6	4	2	1	0.513

Consistency index = 0.004

The pairwise comparisons of four alternatives with respect to age are shown in Table 14. The weight of Iida was highest, because Iida is youngest.

Table 14. Pairwise comparisons of alternatives with respect to age.

	Standridge	Bullington	Mima	Iida	Weight
Standridge	1	1/2	1/4	1/6	0.074
Bullington	2	1	1/2	1/4	0.138
Mima	4	2	1	1/2	0.275
Iida	6	4	2	1	0.513

Consistency index = 0.004

The pairwise comparisons of four alternatives with respect to right/left throw are shown in Table 15. The weight of Iida was highest, because left throw is a few and important for Chiba Lotte Marines.

Table 15. Pairwise comparisons of alternatives with respect to right/left.

	Standridge	Bullington	Mima	Iida	Weight
Standridge	1	1	1	1/2	0.2
Bullington	1	1	1	1/2	0.2
Mima	1	1	1	1/2	0.2
Iida	2	2	2	1	0.4

Consistency index = 0

The pairwise comparisons of four alternatives with respect to feasibility are shown in Table 16. The weights of Standridge and Bullington were highest, because they declared *free agent*.

Table 16. Pairwise comparisons of alternatives with respect to feasibility.

	Standridge	Bullington	Mima	Iida	Weight
Standridge	1	1	3	3	0.375
Bullington	1	1	3	3	0.375
Mima	1/3	1/3	1	1	0.125
Iida	1/3	1/3	1	1	0.125

Consistency index = 0

Table 17 shows final results of AHP. Standridge was the best. Because we assumed that capability and feasibility are most important. Capability and feasibility of Standridge are highest. Standridge is selected as the final choice. Actually, Chiba Lotte Marines acquired Standridge as a reinforced starting pitcher.

Table 17. Final results of AHP.

Criteria	Capability	Salary	Age	Right/left	Feasibility	Result
Weight of criteria	<u>0.362</u>	0.039	0.076	0.161	<u>0.362</u>	
Standridge	<u>0.416</u>	0.074	0.074	0.2	<u>0.375</u>	<u>0.327</u>
Bullington	0.242	0.138	0.138	0.2	0.375	0.271
Mima	0.151	0.275	0.275	0.2	0.125	0.164
Iida	0.191	0.513	0.513	0.4	0.125	0.238

5 Conclusion

We proposed a way of clustering professional baseball players with SOMs, considering several team reinforcement strategies using player maps, and deciding team reinforcement strategy with AHP. We used data of pitchers of Japanese professional baseball teams. We used data of pitchers in Japanese baseball teams. First, we collected data of 302 pitchers and clustered these pitchers using fourteen features. Second, we created pitcher maps of all teams and each team with SOM. Third, we examined main features of each cluster. Fourth, we considered team reinforcement strategies by using pitcher maps. Finally, we used AHP to determine the team reinforcement strategy. In future work, we will apply our way to the other sports such as football and basketball. We will use other types of AHP [7] and ANP [12] for decision making.

References

1. Tolbert, B., Trafalis, T.: Predicting major league baseball championship winners through data mining. Athens J. Sports (2016). <https://www.athensjournals.gr/sports/2016-3-4-1-Tolbert.pdf>
2. Ishii, T.: Using Machine Learning Algorithms to Identify Undervalued Baseball Players (2016). <http://cs229.stanford.edu/proj2016/report/Ishii-UsingMachineLearningAlgorithmsToIdentifyUndervaluedBaseballPlayers-report.pdf>

3. Pane, M.: Trouble with the Curve: Identifying Clusters of MLB Pitchers using Improved Pitch Classification Techniques (2013). <http://repository.cmu.edu/cgi/viewcontent.cgi?article=1184&context=hsshonors>
4. Tung, D.: Data Mining Career Batting Performances in Baseball (2012). <http://vixra.org/pdf/1205.0104v1.pdf>
5. Vazquez Fernandez de Lezeta, M.: Combining Clustering and Time Series for Baseball Forecasting (2014). https://repositorio.uam.es/bitstream/handle/10486/661046/vazquez_fernandez_de_lezeta_miguel_tfg.pdf
6. Kohonen, T.: Self-Organizing Maps. Springer, New York (1995)
7. Saaty, T.: The Analytic Hierarchy Process. McGraw-Hill, New York (1980)
8. Kohara, K., Tsuda, T.: Creating product maps with self-organizing maps for purchase decision making. *Trans. Mach. Learn. Data Min.* **3**(2), 51–66 (2010)
9. Doizoe, J., Kohara, K.: Clustering and visualization of goods with self-organizing maps. In: *Proceedings of 70th Annual Convention of Information Processing Society of Japan*, vol. 4, pp. 911–912 (2008). (in Japanese)
10. NPB (Nippon Professional Baseball Organization). <http://npb.jp/>
11. Professional Baseball Data. <http://baseballdata.jp/>
12. Saaty, T.: The Analytic Network Process. Expert Choice, Arlington (1996)



Data Mining with Digital Fingerprinting - Challenges, Chances, and Novel Application Domains

Matthias Vodel^{1(✉)} and Marc Ritter²

¹ University Computer Centre, Chemnitz University of Technology,
09107 Chemnitz, Germany
vodel@hrz.tu-chemnitz.de

² Professorship Media Informatics, Mittweida University of Applied Sciences,
09648 Mittweida, Germany
ritter@hs-mittweida.de

Abstract. During the last decades, digital fingerprinting was used for hundreds of security-related applications. The main purpose relates to tracking and identification procedures for both users and tasks. The role of digital fingerprinting in data mining area became very important. As a key scale-out technology, *thermal fingerprinting* represents an experimental case study, which was introduced to show new application domains for fingerprinting-based profiling. We are now able to monitor all kind of sensor sources in a generic way. The concept is adoptable to hundreds of novel application domains in the IoT & smart metering context.

In this paper, we summarize key features of the thermal fingerprinting approach. The feasibility is demonstrated in a large scaled data centre testbed with typical sensor sources, e.g., temperature, CPU load behaviour, memory usage, I/O characteristics, and general system information. As a result, the approach generates two-dimensional unique and indexable patterns.

Besides this case study, we introduce several further use cases for this kind of sensor data fingerprinting. This includes data mining projects in the area of urban mobility profiling or innovative & lightweight weather forecast models, but also profiling capabilities in body area networks (health monitoring, fitness applications). Finally, we describe remaining challenges and critical security issues that still have to be solved.

Keywords: Digital fingerprinting · Data mining · Sensor networks
IoT · Profiling · Classification · Sensor data fusion
Digital fingerprint · Security · Monitoring
Experimental application domains

1 Introduction

Digital fingerprinting represents one of the key technologies during the last decades in the context of tracking and identification techniques. One important aspect deals with conventional security scenarios, e.g., digital forensics [1]

or authentication/authorization concepts based on biometric watermarking [2]. Starting from traditional fingerprinting approaches in the domain of OS, browser or canvas tracking [3], straight forward to deeper data mining concepts for identifying and profiling user behaviour in web services [4]. Nowadays, millions of people are using multimedia-fingerprinting apps for identifying music tracks with only one click, e.g., SoundHound or Shazam [5].

However, in the current digital age, thousands of sensor sources are connected via network interfaces and application-specific web-services. Here, several key questions are critical. Which kind of profiling is possible based on these information? How effective are anonymization techniques for personal user data in lifestyle apps, smart products or in the domain of health & fitness online services.

To answer these questions, we started a proof of concept in 2016. In (Vodel and Ritter) [6], our initial goal was to optimize the energy efficiency of our data centre (DC). For the daily operation, thousands of kWh power were wasted for oversized air conditioning. To solve this problem, we began to monitor sensor data from the server infrastructure to predict the thermal impact of software tasks to the DC environment. These sources include system health status, memory consumption, hard disk activity, CPU load, network load, or further system temperature values. In order to extract key features from the data sets, each sensor source was monitored and analyzed individually. Different sets of sensor sources were also merged and processed together. This approach provides excellent capabilities for the identification and tracking of specific software tasks in the DC by analyzing thermal loads and the thermal impact on the system behaviour. Thus, we were able to reduce the necessary amount of energy for cooling our DC environment significantly [6, 7].

In a next step, we introduced thermal fingerprinting in 2017 [7, 8] as a generic concept for sensor data fingerprinting. It represents a toolkit of basic data processing techniques for profiling and tracking of all kinds of given sensor data sources. In order to describe the current research work with respective case studies, the following paper is structured as follows. Section 2 summarizes all the key features of the thermal fingerprinting concept, specific parameters, benefits, and limitations. Section 3 presents our reference scenarios with taking relevant environmental conditions, and the setup for the adaptive learning algorithm into account. Subsequently, Sect. 4 summarizes the respective results and includes a short research discussion. Section 5 introduces the actual research work with three different application scenarios of the approach: *urban mobility profiling*, *weather forecast*, and *profiling based on body area networks*. A final conclusion of the paper clarifies the benefits as well as the risks regarding privacy and security.

2 Concept Basics—Thermal Fingerprinting in a Nutshell

Our thermal fingerprinting approach represents a feasible toolkit for software-based status estimation in heterogeneous hardware environments. We want to provide a cost-efficient solution with no further hardware efforts, which is scalable to different environmental conditions and has a short learning stage/initial setup.

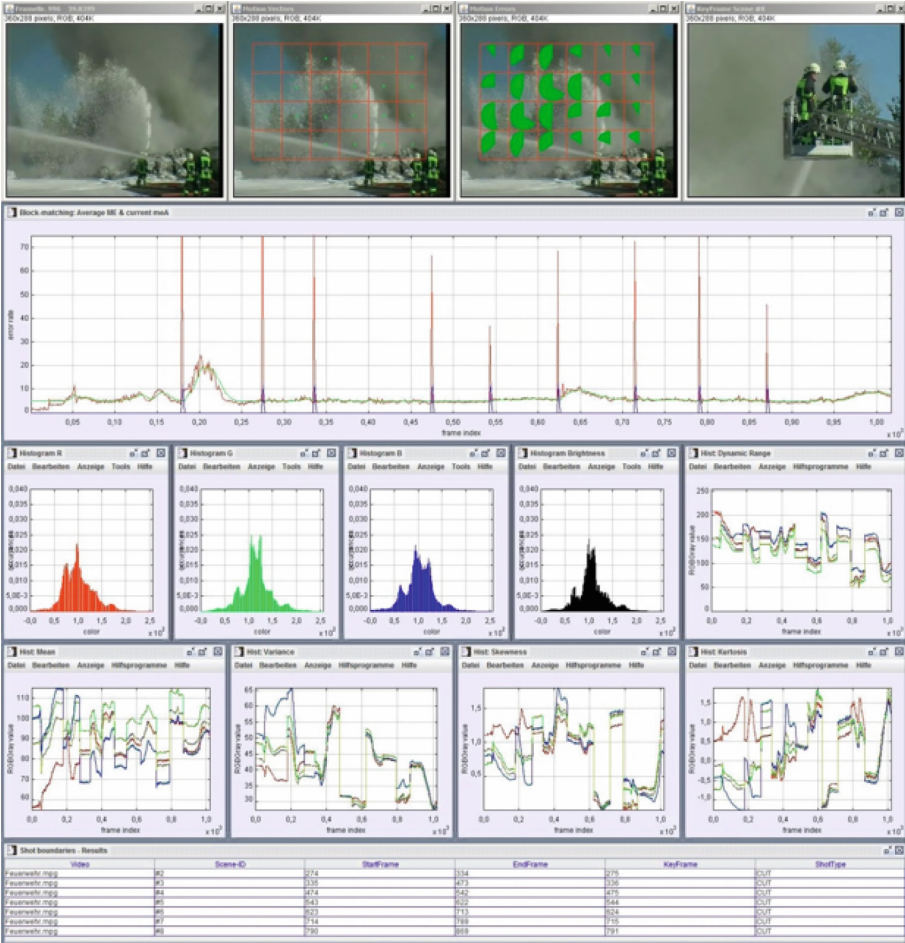


Fig. 1. Screenshot of the used AMOPA framework - an Automated Automated Moving Picture Annotator.

Based on the individual fingerprint patterns and the respective classification features, the expected re-use factor of the generated knowledge appears to be very high.

For the thermal fingerprint approach, we adapted multimedia data processing workflows from the research and teaching framework *AMOPA* (*A*utomated *A*utomated *M*oving *P*icture *A*nnotator) [9,10] (see Fig. 1). Here, the conceptual focus of our approach aims to the demands of computational software task analysis. Accordingly, the system must be capable to handle the respective thermal sensor data originating from different kinds of sensor classes.

2.1 Data Processing Framework

A very important part for this approach is a sufficient software integration and an appropriate design of the corresponding user-interface that facilitates the analysis of pattern by creating opportunities for direct interaction with the data. Our goal is to integrate the entire data processing chain into a single holistic framework. Therefore, we have to modify several input modules of the *AMOPA* framework in order to support the following types of sensor sources (gray represents w.i.p):

- temperature
- CPU load
- I/O load
- Network load
- S.M.A.R.T. information
- acceleration & speed
- positioning data
- inclinometer data
- air pressure
- humidity

A basic operational workflow (single process instance) consists of several processing steps. Each process is derived from a single thread class that works on the input data and stores the resulting data for further processing steps. The chain can be concatenated as well as branched by using XML patterns. The data is automatically transmitted to the next process by the *AMOPA* framework. The adapted input process aggregates the already mentioned sensor types as well as predefined, environmental parameters. One key result of the workflow represents the calculation of two-dimensional fingerprint patterns.

2.2 Thermal Pattern Calculation

In order to calculate two-dimensional, individual patterns, a dedicated data analysis sequence has to be processed. For this purpose, we are working with dynamic sliding window approaches in the time domain, as often used for distributed network simulators [11]. Accordingly, the given sensor data will be merged and analyzed in several time spans in parallel by using different window sizes. Starting from the last 60s and up to 7,200s of logged sensor data, the signal curves are processed. This allows both, the analysis of short term computational load behaviour as well as long term thermal sequences. The workflow is structured as follows being supplemented by Fig. 2.

At first, we define a specific window size. Let the size be for example 120 units, what can be either seconds or minutes. Then we process each data channel of input sensors separately and on each specific window with overlap (e.g., at step size 10) in the top row (left). Here, the input sensors show values of *CPU load* (green), *Temperature* (red), and *CPU relative load* (blue). The current classification of the original computation pattern is illustrated on the right being a set

of *computing* (red), *web services* (yellow), and *backup & maintenance* (green). We used the traffic scheme in order to indicate the computational load of the machine. For each of the obtained windowed data distributions, we apply the following procedure:

1. Subtract the mean from the current data distribution window (top row).
2. Compose (multiply) the data with a sinusoidal wave that works as a carrier function ranging from 0 to 86 in quarter steps of π (middle row).
3. Calculate the histogram from the composed data (bottom row).

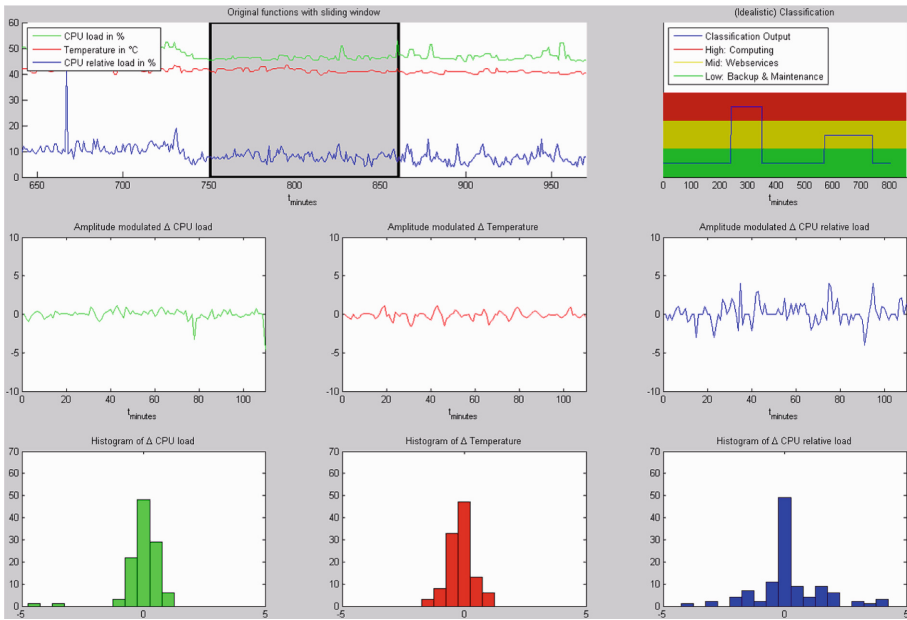


Fig. 2. Example set for thermal fingerprint of backup & maintenance tasks in data centre environments. (Color figure online)

As an intermediary result, we receive specific histograms for different time slots, which represent deviations from the given baseline. The threshold value between the different deviation zones represent the already mentioned risk levels.

In order to stabilize those results, we apply the window data distributions from step 2) to another method. The analysis of spectrograms is very common in the audio domain. The composed/transformed sinusoidal waveform allows us to use such methods in this domain of data. In order to yield reasonable spectrograms, we first have to transform the signal with a butterworth filter that was designed by using the MATLAB DSP Toolbox for highpass filters with the coefficients of filter order 10, a cutoff frequency of 3 dB below the passband value of 25, and a sampling frequency of 200 Hz. For the already mentioned

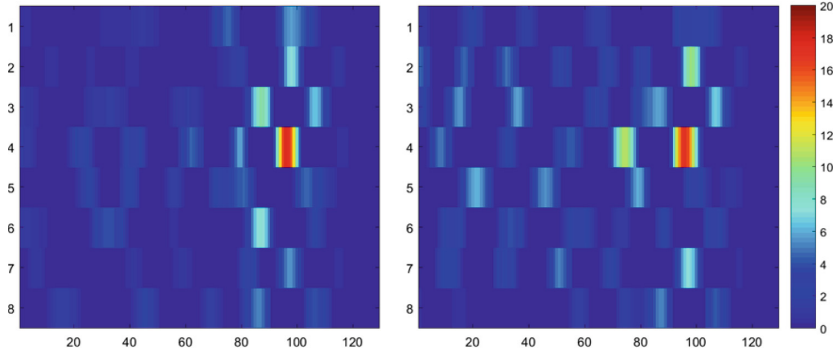


Fig. 3. Example set for thermal fingerprint of the computational task in our data centre environment for the CPU load (left) and the CPU relative load (right). (Color figure online)

example above (120 units), we generate a spectrogram over the whole window with MATLAB standard parameters¹.

By default those parameters divide the window data distribution into 8 segments with 50% overlap using a Hamming window. The number of frequencies points is limited to 128.

The results are two-dimensional, coloured patterns, which represent unique fingerprints. Examples of the acquired spectrogram fingerprints are shown in Fig. 3, providing several kinds of information. On the X axis, the time domain in the sample window sequence of 120 units is given whereas the Y axis contains the corresponding segments. The individual coloured cells describe the signals intensities in pseudo-colours. The figure clarifies higher frequencies (red) in the patterns for CPU load (left) and CPU relative load (right) at around 100 units within this window.

2.3 Fingerprints Storage and Discussion

The pattern can be stored in a database as a simple concatenation of the hexadecimal colour values of each sector. The pattern resolution is predefined with a static number of X columns and Y rows. The fingerprint pattern can be used as a general knowledge base to identify and recognize repeating tasks.

Actual and future research work investigate topics for an efficient classification or categorization of these patterns. Here, the key challenge is represented by the different variations of similar sensor tasks. These challenges are comparable to the already mentioned music and multimedia databases like Shazam or SoundHound. The system must be capable of detecting similar tasks with modified environmental conditions or different initial system states. Accordingly, the patterns provide significant, domain-specific characteristics with scenario-specific,

¹ <http://de.mathworks.com/help/signal/ref/spectrogram.html>, 2016-06-09.

adapted shapes. Hence, our system needs to adapt to such characteristics in a generic way and with minimal training efforts. Please note that the self-learning capabilities as well as the pattern classification are not a focus of this paper representing early work in progress.

3 Reference Scenarios and Test Bench

In order to test the proposed approach under real-world conditions, we have conducted some case studies. We used a large-scaled, heterogeneous data centre testbed, which includes the entire DC infrastructure with multiple virtual and physical components. The testbed is located at the Chemnitz University of Technology and consists of multiple server, storage, and network components. We also evaluated the thermal impact to the environment under real world conditions. Accordingly, the test scenario also includes data sources from the air-conditioning control system as well as from multiple cold aisle containment groups.

3.1 Setup and Environmental Conditions

In order to provide sufficient sensor data, we measured all available temperature sensor inside the individual hardware nodes (CPU, chassis, memory, power supply, and GPU if available). The sampling rate is static and predefined with one measurement per second. Furthermore, the logging system grabs the CPU load, I/O load and network load with the same sampling frequency. The data was stored on a dedicated logging server for the entire post-analysis routines.

For this contribution, we analyzed different hardware from our IT infrastructure. This includes a Cisco UCS chassis for our desktop virtualization, a central storage system (NetApp FAS series) as well as Dell Poweredge servers as dedicated compute nodes as service-providing systems. The proof-of-concept consists of 48 h log data for each system.

The pattern calculation process as well as the data annotation framework will be demonstrated at the conference. Due to security reasons, an external access to our test environment is not possible. But anyway, the conceptual core of this paper represents the generic sensor data processing for generating individual, digital fingerprint patterns. Thus, we are able to profile any kind of sensor components or sensor sources. Based on a scenario-specific requirements set, we adapt each data processing step inside the AMOPA framework.

3.2 Limitations and Constraints

After summarizing the benefits of the approach and the provided opportunities, we furthermore discuss some of its weak aspects. One limitation deals with the given sampling rate for our test scenario of only one measurement per second. Here, a higher data resolution allows a more precise data analysis.

With respect to our test setup, another limitation represents the missing reference values for the temperature measurements. We are using standard system interfaces for extracting the sensor data from the hardware components. These values are not referenced and may differ between similar systems. Only a few global reference temperature sensors inside the data centre testbed are available, which are used as conventional control input for the air-conditioning system.

Further constraints address the mapping between sensor data and software tasks. In order to define the relation between measurement sets and the respective computational tasks, we are using the log files of the systems. Minimal time drifts between different systems are possible but uncritical. The system log correlates with the time stamps of the measurement sets and enables a direct mapping of a system load to the thermal load as well as the respective estimation for the thermal impact.

4 Results Analysis and Scale-Out

Based on the given data centre infrastructure, several systems are monitored and processed. The proposed approach generated individual fingerprints for any type of system (both physical or virtual). Figure 4 illustrates a small number of reference patterns.

The figure includes the following tasks:

1. Top left: Cisco UCS virtualization cluster
→ cluster-internal VM migration task for cluster maintenance preparation
2. Top right: Cisco UCS virtualization cluster
→ boot up sequence for a virtual PC pool of 20 hosts
3. Center left: DELL Poweredge 710 server
→ short term batch <5 min (user-specific computational task)
4. Center right: DELL Poweredge 710 server
→ parallelized long term term batch (about 60min user-specific task with I/O and computational load)
5. Bottom left: NetApp FAS 3240C
→ copy process for 25 GB of user data
6. Bottom right: NetApp FAS 3240C
→ data integrity test for 25 GB of user data

The measured thermal fingerprints are stored in a database knowledge base. This data is used to trigger an adaptive control loop for the air-conditioning system of our data centre. This specific control system is called “TUCool” (TU Chemnitz Cooling Approach) [12]. The system is able to adapt the cooling capacity dynamically, dependent on the system behaviour and given temperature data. Basically, TUCool is working with a predefined or self-learned rule set. The different parameters are used to calculate possible future environmental conditions inside the data centre.

The proposed thermal patterns represent the second stage for this adaptive software-based optimization approach. The fingerprints allow more accurate, faster prediction for the required (near-future) cooling capacity.

If we scale-out the approach to the usage at several locations or institutions, the system allows an open exchange of the fingerprint data. In consequence, an open access database for thermal fingerprints provides the opportunity to minimize efforts for learning the initial setups and schedules. At this point the authors would like to clarify, that such an open access database will be very hard to manage. Due to endless different environmental conditions and operational scenarios, the classification and recognition of known thermal loads at different locations is not that simple.

In discussion of possible worst case scenarios, the following example may summarize the critical points:

Here, a similar computational task would be executed periodically on several comparable hardware platforms but with changing environmental conditions. As a consequence, a sub-optimal data handling results in numberless variations of fingerprint pattern for an identical task. Accordingly, such a scenario generates massive amounts of useless data and complicates the identification/recognition process. Thus, a robust implementation of the proposed core features is essential for a feasible and efficient solution.

5 Research in Progress

After showing the high potential of the presented approach, we are now discussing actual research scenarios, based on the same profiling and classification techniques. The following examples only describes a small number of possible use cases and represent early stages of work in progress.

5.1 Urban Mobility Profiling

Several research projects all over the world are currently analyzing mobility models in urban environments. Townspeople have the choice of using public transportation, individual transports (taxi, etc.), or walk by foot. The decision depends on multiple factors. This includes overall distance, distance to station of public transport, time, day of week, weather and further parameters.

If we take a look on Fig. 5, we see a typical urban topology with different movement sectors. One aim of urban planning and urban management are statistical analysis of how townspeople use different urban transportation offers for different movement routes. In this study, we want to use anonymized positioning data and speed information from mobile phones to identify specific movement characteristics. Four different types of movement are classified:

- walk by foot
- public transportation tram
- public transportation bus
- individual transportation by taxi/car

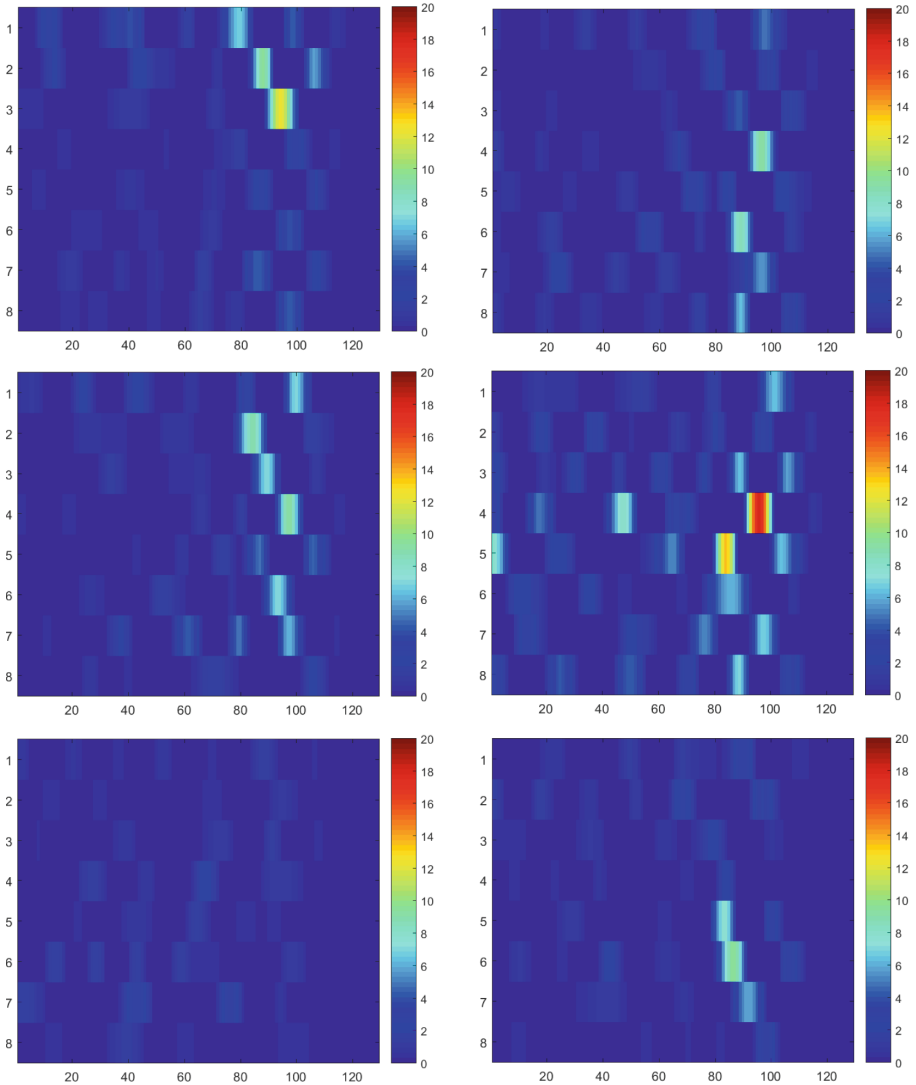


Fig. 4. Six different thermal fingerprints for different types of computational tasks.

Based on these movement classes, we monitor the movement vectors, time-stamps as well as further possible (anonymous) environmental meta information. This study will also clarify the question, if it is possible to identify individual persons based on anonymous urban movement data without any user-specific advance information.

5.2 Weather Forecast

Weather forecast and the prediction of specific meteorologic events represents an entire branch of research. Complex models and enormous computing power are necessary to analyze and process weather data. Figure 6 illustrates these data sources, including hundreds of thousands of wind speed/direction, humidity, air pressure, and temperature data sets. And of course, the results of these weather prediction models are still far away from optimal results.

Accordingly, we plan to use the same key features of the thermal fingerprinting approach to analyze and classify local high level weather events to estimate periodic weather patterns in the near future. The monitoring and pattern matching tasks of only a few, simple weather data sets per location is very easy and does not need that much computational power.

Our research deals with the question, what are the measurable quality differences between traditional weather estimation methods in comparison to this simple, pattern-based forecast model. This question is also very important with respect to the increasing climate changes and more extreme local weather events.



Fig. 5. Typical urban topology for testing the profiling techniques.

5.3 Body Area Network Profiling

A third, very promising research field for the already mentioned thermal fingerprinting capabilities refers to body area network (BAN) applications, e.g., fitness and health monitoring (hard- & software).

As shown in Fig. 7, a typical BAN application consists of several body sensor units for heart beat/pulse, blood pressure, body temperature, pedometer, movement distance, etc. These sensors are connected with a wireless gateway unit, which allows an upload for arbitrary web services.

Several security-critical issues can be summarized. Many interfaces & APIs of the web services are secured very weak and especially the low power communications between sensor and gateway are easy to monitor. Furthermore, a lot of software applications allow the public visibility of fitness profiles or personal health information, maybe with anonymized data sets.

Our current research work in this context deals with concepts for monitoring and classifying body sensor sources for generating user profiles. These profiles can be used for classifying health- and fitness status for specific user groups or specific user parameters (fitness level, health status, restrictions in the human motor unit, etc.). Furthermore, we want to discuss the security impact of these data sets in a critical way. Is it possible to identify individual users behind anonymized fitness- and health data. With respect to the thermal fingerprinting principles, it is also to investigate, which personal user information are calculable/derivable from the user-specific data patterns.

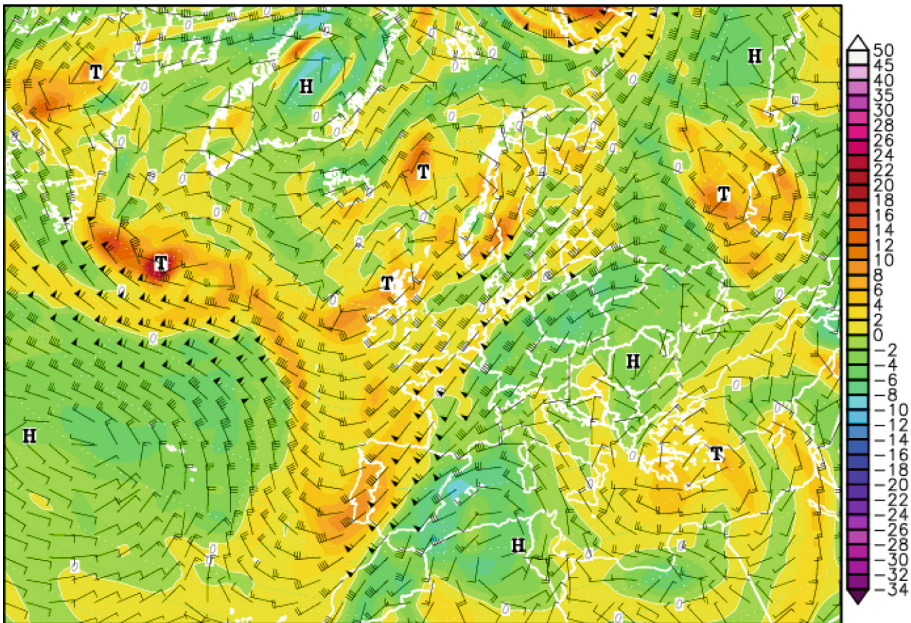


Fig. 6. Weather map with all typical data sources for wind, temperature, and air pressure.

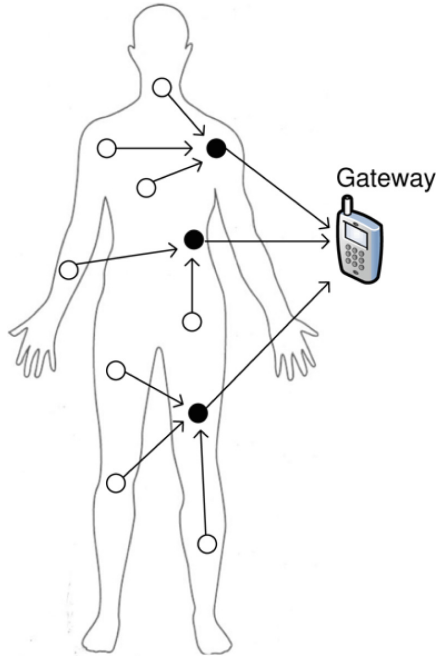


Fig. 7. Body area network topology with multiple sensor sources and wireless monitoring interfaces.

6 Conclusion

In this paper, we introduced a novel approach for data profiling and digital fingerprinting. Based on a large-scaled DC testbed, we clarify the feasibility of an generic fingerprinting concept in the domain of data centre applications. We are able show how software tasks are classified and tracked for possible thermal impacts to the DC environment. Administrators are able to optimize the cooling baseline for the given air-conditioning system without additional hardware efforts. This results in significant savings with regards to energy and money. The used AMOPA data analysis framework as well as the algorithms for the digital fingerprint pattern calculation are robust and flexible for multiple adaptation to a large number of novel, promising applications fields.

Furthermore, we present current and future case studies based on the principles of the thermal fingerprinting approach. Hundreds of promising application domains are possible, e.g., urban transportation optimization, lightweight weather forecast models, or body area network analysis. One key challenge of all these applications still represents issues regarding security & privacy. Modern data mining techniques allow a deep analysis of the merged data and fingerprint patterns.

Acknowledgement. We gratefully acknowledge the cooperation of our project partners and the financial support of the DFG (Deutsche Forschungsgemeinschaft) within the *Federal Cluster of Excellence EXC 1075 “MERGE”*.

References

1. Swaminathan, A., Wu, M., Liu, K.J.: Digital image forensics via intrinsic fingerprints. *IEEE Trans. Inf. Forensics Secur.* **3**(1), 101–117 (2008)
2. Noore, A., Singh, R., Vatsa, M., Houck, M.M.: Enhancing security of fingerprints through contextual biometric watermarking. *Forensic Sci. Int.* **169**(2), 188–194 (2007)
3. Fifield, D., Geana, A., Garcia, M.L., Morbitzer, M., Tyga, J.D.: Remote operating system classification over IPv6. In: *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*, pp. 57–67. ACM (2015)
4. Takei, N., Saito, T., Takasu, K., Yamada, T.: Web browser fingerprinting using only cascading style sheets. In: *10th International Conference on Broadband and Wireless Computing, Communication and Applications*, pp. 57–63. IEEE (2015)
5. Han, B., Hou, Y., Zhao, L., Shen, H.: A filtering method for audio fingerprint based on multiple measurements. In: *Proceedings of the International Conference on Information Technology and Computer Application Engineering*, p. 377. CRC Press, Hong Kong (2015)
6. Vodel, M., Ritter, M.: The TUCool project - low-cost, energy-efficient cooling for conventional data centres. In: *Proceedings of the 6th International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies* (2016)
7. Vodel, M., Ritter, M.: Thermal fingerprinting - multi-dimensional analysis of computational loads. In: *Proceedings of the International Conference on Information Resources Management* (2017)
8. Vodel, M., Ritter, M.: Thermal fingerprints for computational tasks - benefits and security issues. In: *Proceedings of the International Conference on Electronics, Information and Communication* (2017)
9. Ritter, M.: Optimization of algorithms for video analysis: a framework to fit the demands of local television stations. In: *Wissenschaftliche Schriftenreihe Dissertationen der Medieninformatik*, vol. 3, pp. i–xlii, 1–336. Universitätsverlag der Technischen Universität Chemnitz, Germany (2014)
10. Storz, M., Ritter, M., Manthey, R., Lietz, H., Eibl, M.: Annotate. Train. Evaluate. A unified tool for the analysis and visualization of workflows in machine learning applied to object detection. In: Kurosu, M. (ed.) *HCI 2013. LNCS*, vol. 8008, pp. 196–205. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39342-6_22
11. Vodel, M., Sauppe, M., Caspar, M., Hardt, W.: SimANet—A large scalable, distributed simulation framework for ambient networks. *J. Commun.* **3**(7), 11–19 (2008)
12. Vodel, M., Ritter, M., Hardt, W.: Adaptive sensor data fusion for efficient climate control systems. In: Antona, M., Stephanidis, C. (eds.) *UAHCI 2015. LNCS*, vol. 9176, pp. 582–593. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20681-3_55



Categorization of Patient Diseases for Chinese Electronic Health Record Analysis: A Case Study

Junmei Zhong¹(✉), Xiu Yi², De Xuan², and Ying Xie²

¹ Inspur USA Inc., 2010 Suite 150, 156th Ave NE, Bellevue, WA 98052, USA
junmeizhong@inspur.com

² Inspur Software Business Group, Technology Research Center,
1036 Langchao Road, Jinan, China
{yixiu, xuande, xieying}@inspur.com

Abstract. The electronic health record (EHR) analysis has become an increasingly important landing area for machine learning and text mining algorithms to leverage the full potential of the big data for improving human health care. In a lot of our Chinese EHR analysis applications, it is very important to categorize the patients' diseases according to the Chinese national medical coding standard. In this paper, we develop NLP and machine learning algorithms to automatically categorize each patient's diseases into one or more categories. We take each patient's disease description as a document. Also, for each disease category, we make use of its description information in the medical coding standard and take it as a document. According to the characteristics of our data, we define the categorization problem as the unsupervised classification problem with the nearest neighborhood (NN) algorithm using different vector representations to represent the documents. Experimental results show that the averaged word embeddings of word2Vec works best with very promising classification performance.

Keywords: Electronic health record (EHR) analysis · Word embeddings
Text mining

1 Introduction

In Chinese EHR analysis, the categorization of patients' diseases is very important. In some applications, it is necessary to categorize each patient's diseases into different categories such as lung, heart, eyes, ears, nose, liver, and so on. For the massive data set of EHRs, it is desirable to develop some automatic methods to do the categorization. We aggregate the patient's historical diseases scattered in our EHRs, and categorize the individual diseases into one of the 18 categories according to the Chinese national medical coding standard. In the medical coding standard, for each disease category, it has some description information about the related situations and symptoms of the disease, we make use of this category description by taking it as a document and regarding its vector representation as the cluster center of this disease category. Also, we take each patient's symptom description in the EHRs as a document and represent it

with a vector. We develop machine learning algorithms to categorize each patient's disease in a visit or in one treatment by matching the symptom description document to each of the 18 cluster centers, and get the categorization with the highest similarity. A patient may have multiple records for different diseases and his/her whole disease categorizations can be aggregated forming his disease history.

There are two basic components in this system to accomplish the disease categorization. The first one is the NLP algorithms for document tokenization and representation with a vector, and the second one is the machine learning algorithm for disease categorization. For document's vector representation after being tokenized, we have tried the vector space model with both TF-IDF weighting and binary representations for tokens, the averaged word embeddings of word2Vec [1], and doc2Vec [2]. For classification with machine learning, according to the characteristics of our data sets, we select the NN classification algorithm [3] of unsupervised machine learning in this POC project to tackle the challenges. We use the quantitative cosine similarity metric to measure the similarity between a patient's symptom description and one of the 18 disease category descriptions in the medical coding standard for categorization. We experimented with different document's vector representations. The results show that the averaged word embeddings can capture the semantic information in our short document and achieve very promising categorization performance. For demonstrating the application of the categorizations, we make use of the information that in the medical coding standard, each disease category is associated with a specific position or organ on the human body, thus we map each of the patient's disease categorization results to the corresponding human body position or organ, generating a digital patient. This digital patient can provide great help for health care providers to easily understand each patient's whole health situation to make effective diagnosis, treatment and health care plans.

Our contributions lie in the following aspects:

- We define the disease categorization problem for EHRs according to the domain knowledge of medical informatics and Chinese national medical coding standard. This makes it possible to leverage insights from the big EHR data using artificial intelligence (AI) algorithms.
- We develop AI algorithms including NLP, and unsupervised machine learning to tackle the challenges in our data for disease categorization and map the categorization results to the human body picture generating the digital patient for visualization.

The rest of the paper is organized as follows. In Sect. 2 we discuss the methodology about problem definition, vector representations for documents, and the unsupervised machine learning algorithm. In Sect. 3, we present the experimental results. We conclude the paper with some discussions and future work in Sect. 4.

2 Methodology

The patient disease categorization system consists of 3 components. Problem definition and data preparation, NLP for feature extraction, and the unsupervised machine learning algorithm with the nearest neighborhood (NN) classification method for categorizing each symptom description into one of the 18 disease categories.

2.1 Problem Definition and Data Preparation

The Chinese EHRs are from some big hospitals in Shandong Province, China, and they consist of 179 heterogeneous tables such as the patients' personal information and their family information, admission records, hospitalization records, all kinds of lab tests, office visit records, hospital and doctor information, medication information, and surgery information. But in China, creating the EHR for patients is still at the beginning stage, in most of the time, doctors are more willing to take paper notes for patients rather than leave their notes in the computer system, causing the data quality of the EHRs far from being satisfactory. As a result, for the 179 tables, many of them do not contain useful information for a lot of empty columns with the useful information missed. After some preprocessing, only 85 tables are selected. Even in these 85 tables, the useful information is still missed in a lot of records. Through information fusion on the heterogeneous 85 tables with the domain knowledge of medical informatics, we construct individual clinic records about each patient's individual office visits and hospitalizations with individual symptom descriptions, diagnosis records, lab tests, and the doctor's treatment plans and prescriptions for every clinic event. But for this specific categorization task, we only extract each patient's disease symptom descriptions from the constructed clinic records, and take them as documents. Also, we find out that in the medical coding standard, for each disease category, it lists some symptoms from different aspects about this specific disease, so we aggregate them and take the 18 disease categories' descriptions in the medical coding standard as 18 documents. But the attention to be paid is that for both doctors' notes about patients' symptom descriptions and the 18 disease categories' descriptions, they are not composed of regular sentences with syntactic information, but they are a list of words about the symptoms. Furthermore, our statistics show that the EHRs distribute in a very unbalanced way on the 18 disease categories. The records are mostly concentrated on the popular diseases. Knowing the characteristics of our data and with all aggregated information, we define the disease categorization problem as a matching problem from each of a patient's symptom descriptions to the 18 cluster centers. For this, each document is represented as a vector, and the nearest neighborhood classification algorithm is selected to do the categorization with the cosine similarity to quantitatively measure how similar the two documents are.

2.2 NLP for Feature Extraction

In machine learning for document categorization, documents need to be represented as feature vectors, so we first tokenize each unstructured Chinese document into a collection of terms. Since there are no spaces between the Chinese words in each sentence,

which is very different from the English texts that can be tokenized by spaces, the tokenization of Chinese texts needs specific algorithms. Here we only use the available tool for this task and focus our efforts on the other things based on this tokenization result. The Han LP, an open source software is used for tokenizing the Chinese documents, then we generate a structured vector representation for each document with 4 different ways: the vector space model with TF-IDF weighting method and the binary representation for tokens, the averaged word embeddings of word2Vec, and document vector of doc2Vec.

2.2.1 The Bag of Words (BOW) Method

The BOW method makes use of the tokenized individual words and/or N-Grams ($N \geq 1$) in a corpus as features for a document’s vector representation, which is usually called a feature vector in machine learning and pattern recognition. All N-Grams constitute the vocabulary of the corpus and the length of the BOW vector is the size of the vocabulary. If N is equal to 1, the N-Gram is called unigram. For individual tokens, we usually have both binary representation and TF-IDF weighting representation to get the feature values. The binary representation only considers the presence and absence of the individual words in the documents without considering the number of occurrences of them. If a token is present in the document, the vector’s corresponding feature value is 1, otherwise 0. On the other hand, the TF-IDF weighting method takes the product of two statistics, the term frequency and inverse document frequency. The term frequency is simply the number of times that the term t appears in a document d . It assumes that the more frequent the token appears in the document, the more important it is for describing the topics of the document, and it is usually calculated in the following augmented way [4]:

$$tf(t, d) = 0.5 + 0.5 * \frac{f_{t,d}}{\max\{f_{t',d} : t' \in d\}} \tag{1}$$

here $f_{t,d}$ denotes the frequency of term t in document d . At the same time, the inverse document frequency is calculated in the following way [4]:

$$idf(t, D) = \log\left(\frac{N}{|\{d \in D : t \in d\}|} + 1\right) \tag{2}$$

With

- N the total number of documents in the corpus, $N = |D|$
- The denominator $|\{d \in D : t \in d\}|$ is the number of documents where the term t appears. If the term t does not occur in the corpus, the denominator needs to be adjusted into $|\{d \in D : t \in d\}| + 1$.

The inverse document frequency is used to offset the impact of common words in the corpus without having specialty. But the BOW method usually has the following limitations:

- Does not take the order information of words in each sentence into account and only considers the occurrence of a word independent of the others, which is not true from both semantic and syntactic point of view. So, different documents for different topics are easily taken to be similar only if they contain the same words.
- Sparsity and high dimensionality of the feature vectors. Corpora generally have at least thousands of words (features). When 2-grams and 3-grams are considered, the number of features per corpus increases significantly. It could generate a very sparse term-document matrix with a lot of unwanted zeros. Not all features are important for learning problems, and modeling such high-dimensionality data set needs a huge number of annotated samples for training, and it tends to lead to the overfitting problem for supervised machine learning when no sufficient annotated samples are provided. The high dimensionality problem challenges very much training supervised machine learning models for the curse of dimensionality, and in most of the time, we need to do dimensionality reduction for the BOW vector representations, but it is not a trivial work in preserving the structural information when reducing the dimensionality.

2.2.2 Word2Vec for Word Embeddings

Since the BOW vector representation usually cannot capture the semantic information from documents with limitations of both high dimensionality and sparsity, researchers have investigated different ways to represent documents and words in an embedded low-dimensional dense vector space. The Word2vec algorithm [1] is such a distributed representation learning algorithm to learn the continuous dense vector representations for words in an embedded low dimensional vector space. It consists of 2 related models: the continuous bag-of-words (CBOW) model and the skip-gram model as shown in Fig. 1. The CBOW model is used to predict the current word from its surrounding context words in a sentence within a window centered at the current word, while the skip-gram model is used to predict the surrounding context words in a sentence within a symmetric window for a given word.

The Word2vec model can be trained with either the hierarchical softmax or negative sampling method. The hierarchical softmax uses a Huffman tree to reduce the computational complexity by only updating the vectors of the nodes on the path from the tree root to the leaf node (the current word), while the negative sampling accomplishes this goal and improves the vector quality of low-frequency words by only sampling a small number of the negative samples for updating the vectors in the backpropagation process for which the high-frequency words are down-sampled and the low-frequency words are up-sampled by frequency lifting. These models are the two-layer shallow neural networks. Word2vec takes as its input the high dimensional one-hot vectors of the words in the corpus and produces a vector space of several hundred dimensions such that each unique word in the corpus is represented by a dense vector in the vector space. A very salient feature of this kind of word embeddings is that word vectors in the vector space are close to each other when the words are semantically similar to each other. This offers the benefit that we can infer semantically similar words from the vector space for a word if the word's vector is known and it hence has attracted tremendous attention in text mining, machine translation, and

document analysis. But the word2Vec algorithm still has its limitation in representing documents. The usual way of using the word vectors for document classification and clustering analysis is to take the averaged word vectors or other weighting methods for word embeddings in a document.

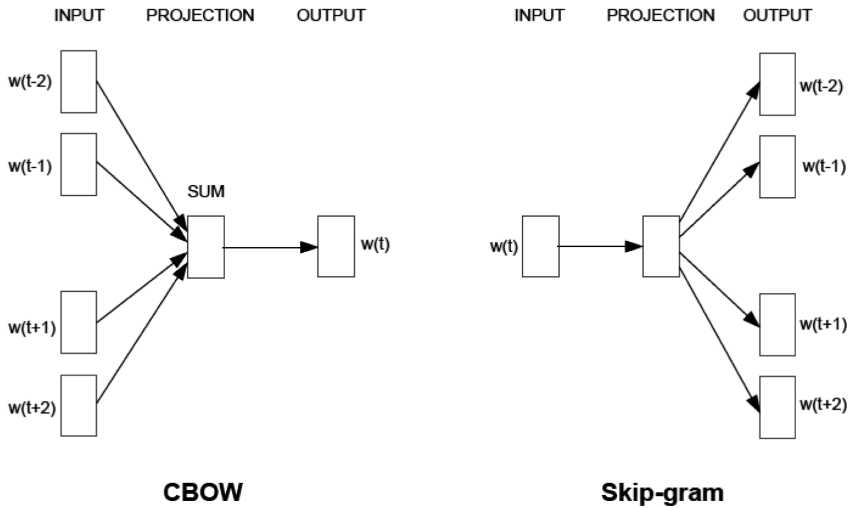


Fig. 1. The CBOW model (left) and Skip-gram (right) model in word2Vec with courtesy of Mikolov et al. [1].

2.2.3 Doc2Vec

The Doc2Vec algorithm [2] is an extension of Word2Vec that tries to represent a document or paragraph as a single real-valued dense vector in addition to generating the word vectors for individual words in the corpus. In training the doc2Vec model, each document is assigned a unique tag together with the other words/terms in the document, and finally each unique document tag is assigned a dense vector. Just like word vectors generated by word2Vec, which can provide semantic inference for individual words, a document vector generated by doc2Vec can be thought of reflecting some semantic and topic information of the document. As a result, document vectors of similar documents tend to be close to each other in the embedded vector space. It is very useful for document classification or clustering analysis with the generated single document vector. Applications include sentiment analysis for movie reviews, and text classifications.

2.3 Unsupervised Machine Learning for Categorizing Diseases

For machine learning, there are generally two kinds of models. One is the supervised learning and the other is the unsupervised learning. For supervised learning, many annotated samples are needed to prevent overfitting. In addition, as the dimensionality increases, the required annotated samples increase exponentially for the curse of

dimensionality. For medical data, this annotation process usually needs the extra supervision from medical experts, making it practically not affordable for our high dimensional data analysis. Furthermore, the records in our EHRs do not distribute uniformly or approximately uniformly on all disease categories, but they distribute in a very spiky way. They are mostly concentrated on the popular diseases, leaving only a small number of records or even a few records for the unpopular diseases. Thus, this kind of unbalanced high dimensional feature vectors is not suitable for supervised machine learning. In this paper, we select the unsupervised machine learning to circumvent the challenges in categorizing patients' diseases. The benefit of using unsupervised learning is that it does not depend on the distribution of the data in different disease categories once the cluster centers for the 18 disease categories are available. Since we can find some disease description data for each of the 18 disease categories in the Chinese national medical coding standard, we take it as a document and take the document's vector representation as the corresponding cluster center. Thus, after generating a vector representation for the individual documents (patients' diseases), we can accomplish the categorization by using the NN classification algorithm [3]. For this, we choose the quantitative metric of cosine similarity to measure the matching quality between a document and one of the 18 categories' cluster centers. The higher the similarity metric, the more similar the two involved documents are. The quantitative normalized cosine similarity between any two vectors A and B is calculated in the following way [5]:

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\|_2 \|B\|_2} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}} \quad (3)$$

For each patient's document of symptom description, we categorize it into one of the 18 disease categories as shown in Fig. 2. The category's description having the highest cosine similarity metric with the patient's symptom description is assigned as the patient's disease categorization result.

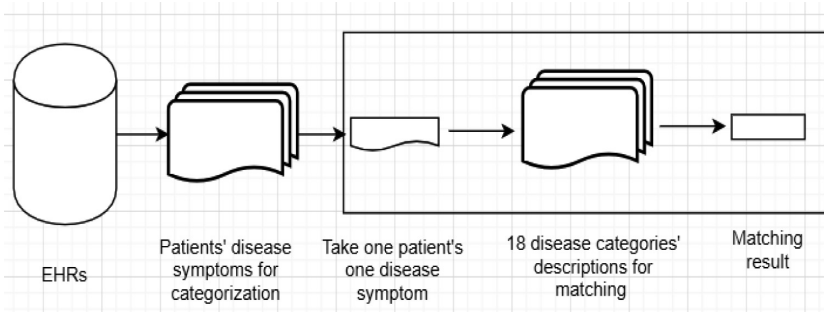


Fig. 2. The diagram of the document matching process for disease categorization.

3 Experimental Results and Analysis

We have investigated two kinds of vector representations for documents. They are the BOW vectors and the word embeddings. For the BOW vectors, they are the TF-IDF vectors and binary vectors. For the word embeddings, they are the averaged vectors of the pre-trained word embeddings of word2Vec, and the vectors of doc2Vec. Totally we have tried 4 document representations. For the TF-IDF and binary vector representations, the vector dimension is 53039 after some preprocessing to remove the stop words and low frequency words. For optimizing the word2Vec, we have tried 4 models:

- CBOW with hierarchical softmax
- CBOW with negative sampling
- Skip-gram with hierarchical softmax
- Skip-gram with negative sampling.

Finally, the CBOW model with negative sampling is selected with the optimized hyperparameters: window size is 3, dimension of embeddings is 200, low frequency threshold for sampling is $1e-5$, 5 samples are used for negative sampling, and iteration number is set 10. The doc2Vec does not work for our data. Its performance is far below that of the word2Vec and it is not listed here. So only the word2Vec is used as the embeddings for document classification. In this paper, the accuracy is used as the quantitative metric for model's performance evaluation and it is defined as follows [6].

$$accuracy = \frac{\text{Number of items of category identified}}{\text{Total number of items of the category}} \quad (4)$$

From experimental results as shown in Tables 1, 2, 3 and 4 for the popular disease categories in our EHRs with sufficient testing samples to calculate the reliable accuracy metric, we can see that the two BOW vectors do not work well, and the averaged word embeddings from word2Vec works best. This is mainly for the two facts: The medical documents about the symptom descriptions are comparatively short. There are some discrepancies for the words used by different doctors to describe the similar symptoms, as a result, the two BOW vector representations cannot capture the semantic information about the symptom descriptions and they generate very high dimensional and very sparse vectors for documents. When they are used for matching two similar documents, they work well if the two documents use the same words for symptom description, otherwise they do not work. As a result, their performance is not very satisfactory. We have also tried to reduce the dimensionality for the TF-IDF vectors and the binary BOW vectors with the principal component analysis (PCA) method [7], but the categorization is even worse. Analysis reveals that it is mainly for the fact that the PCA is a linear transformation and the projection of the sparse matrix obtained from short documents, to the selected eigen space results in a lot of information loss. Furthermore, as shown in Tables 2, 3 and 4, both TF-IDF and binary vector representations work similarly. When we set a threshold for the term frequency above 15 to get

the binary vector representation for each document, the classification accuracy is a little bit different, otherwise the classification accuracy is the same as that of TF-IDF vector representation. As for the doc2Vec, our analysis reveals that its inefficiency is mainly caused by a few facts. The doc2Vec algorithm is essentially an extension of the word2Vec algorithm with an added document tag for each document and it tries to extract and summarize the semantic information of the sentence or paragraph or document. But each short symptom description in our EHRs mostly consists of individual items and facts about the patient's disease from different aspects without containing any syntactic information like the movie review sentences used for training the doc2Vec model, as a result, the generated vectors for the document tags cannot capture much semantic and topic information from our short medical documents. On the other hand, the word embeddings of word2Vec could provide much more semantic information for words in our short symptom descriptions because the word2Vec algorithm can be pre-trained using a bigger medical corpus beyond the medical symptom descriptions. Normally, the averaged word embeddings of the words in a document will deviate very much from the main topic of the document due to the introduced noise, however, when each disease category's description is composed of a list of symptoms for describing different aspects of the same disease, the words in the description will be semantically related to each other, so their vectors in the embedded vector space tend to be close to each other, and the averaged word embeddings can still reflect their semantic information. The same situation holds for the averaged word embeddings for each patient's symptom description because they are not the usual sentences. As a result, it is natural for the averaged word embeddings to accomplish the best disease categorization result among all vector representations according to the prediction accuracy.

To visualize the disease categorization result, we create a digital patient as shown in Fig. 3, in which a patient's own disease categorizations and his/her family's genetic disease categorizations are mapped to the corresponding human body positions and organs, respectively for visualization. In Fig. 3, the left side demonstrates a patient's 3 diseases with red points on a 2-dimensional human body picture while the right side demonstrates his/her risks from family's genetic diseases including the heart disease with yellow points on a 2-dimensional human body picture. In the future, we are going to create a 3-dimensional human body structure picture, forming the holographic digital patient. Also, some important information about the patient is displayed in the most-left side below the patient's picture, for example, the patient's history of medication allergy, and surgery history. The benefits of this digital patient can help health care providers to immediately understand the patient's health situations and his/her potential risks from family history of genetic diseases with only a glance at the pictures without spending too much time to read the lengthy notes in the records. This can greatly save the doctors' time in making personalized and effective treatment plans. The NLP and machine learning algorithms are implemented in Python, and the open source software of machine learning library Scikit-learn is used.

Table 1. The categorization accuracy for the averaged word2Vec for documents.

Organ/category	Accuracy
Eye	0.926606
Ear	0.871287
Nose	0.940171
Male	0.942789
Female	0.984155

Table 2. The categorization accuracy for the TF-IDF vector representation for documents.

Organ/category	Accuracy
Eye	1.000000
Ear	0.465347
Nose	0.521368
Male disease	0.401669
Female disease	0.588339

Table 3. The categorization accuracy for the binary vector representation for documents with the threshold for term frequency < 15 .

Organ/category	Accuracy
Eye	1.000000
Ear	0.465347
Nose	0.521368
Male	0.401669
Female	0.588339

Table 4. The categorization accuracy for the binary vector representation for documents with the threshold for term frequency ≥ 15 .

Organ/category	Accuracy
Eye	1.000000
Ear	0.465347
Nose	0.521821
Male	0.401669
Female	0.588339

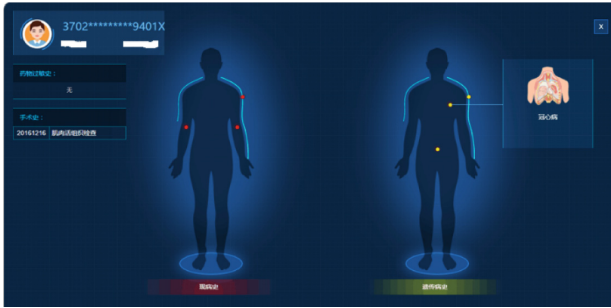


Fig. 3. The digital patient: One application of visualizing the disease categorization results for a patient (left) and his family (right). (Color figure online)

4 Conclusion and Future Work

In this paper, we develop NLP and unsupervised machine learning algorithms to categorize patients' diseases into 18 standard categories as a POC project according to the Chinese national medical coding standard. Through experiments with different vector representations for documents, we find out that for documents in lack of syntactic and grammatical information, effective semantic vector representation plays a significant role in document categorization and the averaged word embeddings accomplishes very promising results. In the next step, we are going to further investigate other vector representations with deep learning architectures such as convolutional neural networks (CNN) and long short time memory (LSTM) networks to see if better semantic vector representations can be obtained for further improving the accuracy in categorizing patient disease descriptions. Also, we are extending this categorization method to International Statistical Classification of Diseases and Related Health Problems (ICD)-10. At the same time, we have provided our findings and suggestions to the Chinese Health Planning Commission about how to make the hospitals provide high-quality EHRs to benefit the society.



Acknowledgments. The authors would like to thank the support from both the Technology Research Center and the Business Unit of Medical Health of the Software Business Group, Inspur Inc, and the Health Planning Commission of Shandong Province, China.

References

1. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. *Adv. Neural Inf. Process. Syst.* 3111–3119 (2013)
2. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: *International Conference on Machine Learning*, pp. 1188–1196 (2014)
3. Aas, K., Eikvil, L.: *Text categorization: a survey* (1999)
4. Wikipedia Homepage. <https://en.wikipedia.org/wiki/Tf-idf>
5. Wikipedia Homepage. https://en.wikipedia.org/wiki/Cosine_similarity
6. Wikipedia Homepage. https://en.wikipedia.org/wiki/F1_score
7. Abdi, H.: Principal component analysis. *Comput. Stat.* **2**, 433–459 (2010)



Dynamic Classifier and Sensor Using Small Memory Buffers

R. Gelbard^(✉)  and A. Khalemsky^(✉) 

Information Systems, Bar-Ilan University, Ramat-Gan 5290002, Israel
gelbardr@mail.biu.ac.il, anna.khalemsky@gmail.com

Abstract. A model presented in current paper designed for dynamic classifying of real time cases received in a stream of big sensing data. The model comprises multiple remote autonomous sensing systems; each generates a classification scheme comprising a plurality of parameters. The classification engine of each sensing system is based on small data buffers, which include a limited set of “representative” cases for each class (case-buffers). Upon receiving a new case, the sensing system determines whether it may be classified into an existing class or it should evoke a change in the classification scheme. Based on a threshold of segmentation error parameter, one or more case-buffers are dynamically regrouped into a new composition of buffers, according to a criterion of segmentation quality.

Keywords: Dynamic classifier · Dynamic rules · Big data · Sensing data
Memory buffers · Clustering · Classification

1 Introduction

Sensors are located in environments that change dynamically and are required not only to detect the values of all parameters measured, but also to assess the situation and alert accordingly, based on predefined rules managed by a “Classifier-engine”. Since the environments are dynamically changed and there may be new situations that were not known in advance, which are reflected in new combinations of parameter values, there is a need for a dynamic updating of the sensor’s classifier.

One exemplary application for such dynamic classification unit (DCU) is a screening gate including biometric sensors that screen travelers entering a high security area such as an airport. The sensors may be configured to test multiple parameters of a traveler, such as heart rate, heart pressure, perspiration, etc. The classification system may be set to measure two classes of travelers, the bulk of travelers who have “normal” parameters and should pass the biometric screening without interference, and those who should be checked by security personnel. Upon receiving a new case, the sensing system determines whether it may be classified into one of the existing classes, or it should evoke a change in the classification scheme. Thus, over the course of a day, environmental conditions may change; ranges of values that haven’t been observed before may appear causing dynamic changes-updates in sensor’s classifier.

Changes in the sensor’s classifier (i.e. classification scheme) are triggered based on a threshold of segmentation error parameter. The sensor’s classifier is based on small

data buffers and collects remembers a limited set of “representative” cases for each class (case-buffers). As a result of the trigger’s appeal, one or more case-buffers are dynamically regrouped into a new composition of buffers, according to segmentation quality criteria.

The novelty of this real-time mechanism lies in the fact that the entire process is based on the use of limited memory buffers. In addition, each DCU, which is a remote autonomous sensing system, can communicate with multiple additional remote autonomous sensing systems. In such situations, the case buffers, as well as the case history, can be synchronized and managed via a central controller. Furthermore, in a distributed environment, regardless the existence of a central controller, the contents of the case buffers and the classifier scheme of each DCU can be synchronized between the multiple remote autonomous agents (sensing-systems). Synchronization may be performed after each regrouping process. That is to say that each incremental updating at any local DCU may initiate synchronization among all connected autonomous agents.

2 Related Work

In the reality of the dynamic data environment, when a huge amount of raw data and information flows ceaselessly, the main purpose of individuals and organizations is discovering the optimal way to find a hidden potential in it, through the constant cooperation of human intelligence and machine capabilities. The techniques and models that successfully functioned in stable data environment are outdated and need to be corrected to deal with dynamic data environment. “Databases are growing in size to a stage where traditional techniques for analysis and visualization of the data are breaking down” [1, 2]. Because of the constant increase in data volume, interpreting of similarities of different sub-populations becomes the new dimension of data mining goal. The data usually flows from different sources and has to be handled and processed simultaneously [3]. The development of new and advanced techniques in data mining in dynamic data environment covers more and more fields, for instance, computer sciences, medicine [4], security systems [5] and social networks [6, 7]. And it is not just an application of existing algorithmic tools in these fields, but the inclusion of elements and logic and even tools, that were created purposefully for them.

As a result of the constant need to get real-time solutions, the research is naturally directed into a new field – incremental data processing. The motivation is to maximize the quality of solutions through minimizing the process cost [8–10]. The algorithmic tools have to be adjusted to dynamic data environment and be capable to absorb significant amounts of data, possibly to handle with the Big Data environment. The main idea of incremental techniques is to use small segments of data and not the whole historical data [11–13].

One of the commonly used directions in data mining is classification process, in which the objects are classified into homogeneous groups, with a maximal diversity between groups and minimal within groups. The proximity of an object to group centroid is usually measured by similarity measures, such as RMSE (root mean square error), used in the current paper [14]. The classification tasks are usually divided into two main types: if the target attribute is previously known, the process is called “classification”, and if the

target attribute is not known, it is called “clustering” [2, 10, 15, 16]. In the case of clustering problems, the interpretation of achieved clusters is one of the main challenges. For example, if a higher education consultant has to recommend the future student what is the best faculty for him, he will probably pick the faculty name from the existing list in the university. On the other hand, if the security system bank controller needs to identify the type of a new financial fraud trend, he needs to be very open-minded and be able to classify the action undertaken by a fraudster to an absolutely different type and give it an appropriate description. In some cases, there is a need to create a set of groups/classes based on items/customers/actions that are needed to be classified without any information about the target attribute. Different kinds of classification/clustering tasks in dynamic data environment in combination of existing and new techniques became the basis for extensive research [17–19].

The current research presents a dynamic classifier based on incremental dynamic clustering process. It permits the use of small data buffers that represent existing groups. This approach is significantly different from other approaches and methods, considered in the literature.

The dynamic classifier, proposed in the current paper, functions as a sensor. It works as a screening gate that distinguishes between “regular” items that are close enough to at least one of existing groups and alerts when the relatively “different” item occurs. The model permits not just an alert in such situation, but the action required to classify the item. Lots of studies combine different sensors in decision making processes [20–23].

3 Model Architecture

Figure 1 presents a schematic architecture of the proposed system, showing one DCU connected to a central controller as well as to other remote autonomous sensors-agents.

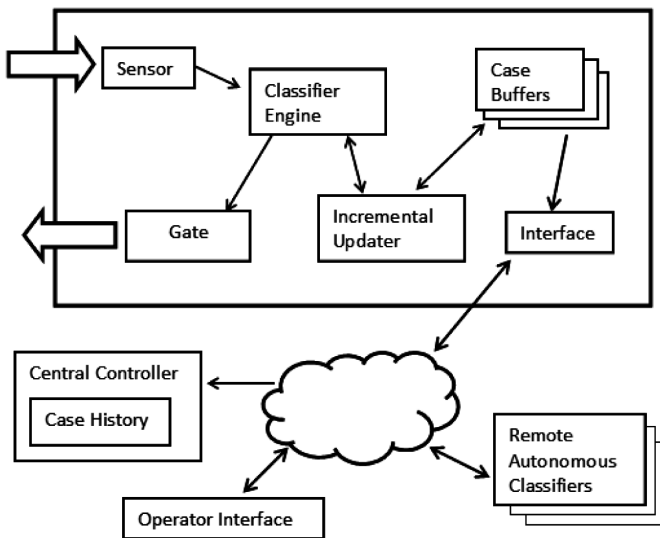


Fig. 1. Schematic architecture of the system.

The real-time data flows through the “Sensor” (Fig. 1, first component) to the “Classifier-engine”. The “Classifier-engine” performs a decision process based on a classification scheme, as described in the flow diagram in Fig. 2. The “Gate” component represents the output, or, in other words, the decision regarding the classification of object. Based on a threshold of segmentation error parameter and segmentation quality criteria, the DCU incrementally updates the population of the relevant “Case-buffers”. The mechanism that manages the population (i.e. cases) stored in each buffer can use diverse policies, such as FIFO policy (First-In First-Out), or a selection policy that may store extreme-farthest cases of each group (“outliers” that are still classified to that group).

The term “Sensor” represents the “funnel” through which the data stream flows. Thus, a sensor can be a physical object, as well as a logical handshake through which the data flows into the system. The flow chart, illustrated in Fig. 2, presents the real-time decision-making process for each new sensing data element (i.e. each new case).

The version control is managed by sequential numbering approach.

Table 1 presents the notations used in the flowchart:

Table 1. Notations

X_i	New case
R_s	The closest group
e	Minimal distance measure RMSE between a new case and existing groups centroids
δ	Threshold that determines the decision of update
R	New group
$R_s(n)$	Number of cases in the buffer
Z_{min}	Minimal number of cases in buffer that justifies the update of the buffer
Z_{max}	The buffer size – maximal number of cases stored in the buffer

The mechanism presented in Fig. 2 works as follows: The new item X_i passes through the sensor, the distance measure between the new item and the centroids of existing groups are calculated and the minimal value e is registered. The threshold level δ , the maximal buffer size Z_{max} and the rest of parameters have to be determined at this point of time. If the minimal distance e is less than a threshold, no rearrangement needed and a new case X_i joins the closest group (completion). If e exceeds the threshold level, the number of items in the buffer is checked. If there are not enough cases (the number of cases is less than Z_{min}), the new case creates an absolutely new group. If there are enough cases in the buffer, the new case removes the oldest case and a new distribution is created (by splitting or merging of the existing groups).

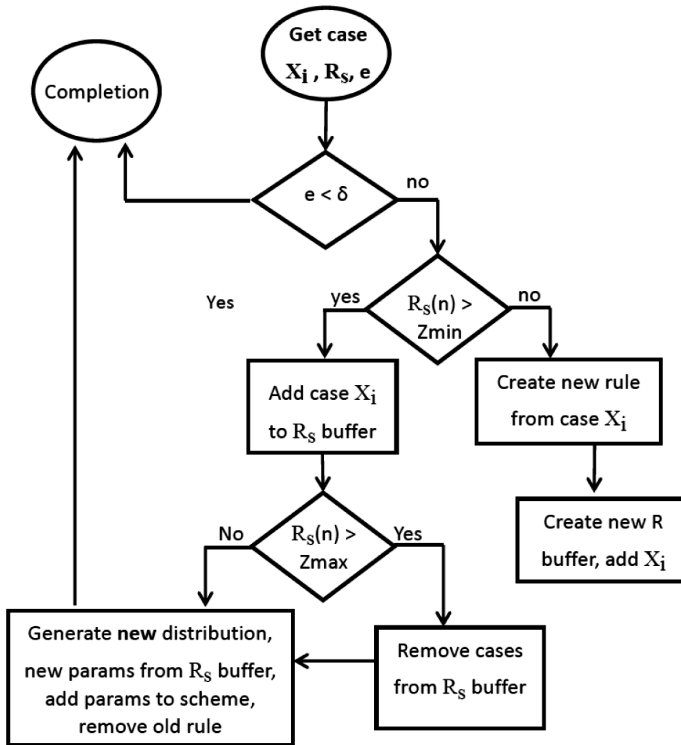


Fig. 2. A flow diagram of a process for real-time data classification based on dynamic updating of sensor's classifiers.

4 Model Validation

Since the model deals with a stream of real-time data, which is a continuous flow of new cases, the validation was based on datasets of classification problems. The model is implemented by the code developed in Python and combines the k-means algorithm package [24].

The following datasets were used: (1) "ERA" dataset, donated by Prof. Ben-David [25]. This data set was originally gathered during the academic decision-making experiment. Input attributes are candidates' characteristics (such as past experience, verbal skills etc.), output attribute is a subjective judgement of a decision-maker to which degree he/she tends to accept the applicant to the job or to reject him. All the input and output attributes have ordinal values. The data set contains 1000 instances, four input attributes and one output attribute. (2) "Car Evaluation" dataset that was retrieved from the UCI Machine Learning Repository [26–28]. Input attributes are cars properties and an output attribute is a class value (unacceptable, acceptable, good and very good). The data set contains 1728 instances.

4.1 Optimal Situation as a Baseline

The theoretical optima in such case is the situation in which the algorithm runs across the entire dataset. Thus, based on the results obtained by the clustering k-means algorithm, while analyzing all the records in the dataset, we can find the best set of rules, and the required total number of rules, that achieve the best classification accuracy.

4.2 The Initial Stage

According to widely used methodology in machine learning, each dataset was divided into training set (with about two thirds of data) and test set (with about one third of data). The training set provides the initial groups and the test set simulates a new data stream. Worth mention that the initial stage is mainly used to shorten the “reset-cycle” of the decision-making process. In cases where there is no urgency, the system can start with no decision rule at all, and with totally empty “Case-buffers”.

4.3 The “Dynamic-Flow” Stage

The test set was used in an unsupervised mode (while hiding the target-labeled field). The records flowed through the “Sensor” to the “Classifier-engine” without any information regarding the right classification-filtering.

- The “Classifier-engine” and the “Incremental-updater” used the flow diagram mechanism described in Fig. 2.
- The delta symbol (δ), in Fig. 2, represents Root-mean-square error (RMSE) that was used as a threshold.
- The parameters in each experiment were set as follows:
ERA data-set: three threshold levels: 2, 2.25, 2.5; initial number of groups: 10; buffer size: 25; training set = 600, test set = 400.

Car evaluation data-set: three threshold levels: 0.8, 0.9, 1; initial number of groups: 15; buffer size: 25; training set = 1400, test set = 328.

In accordance with the schematic architecture of the system (illustrated in Fig. 1), a case is either directly classified, or initiates an incremental reevaluation (supported by the “Incremental-updater” component) till the threshold is satisfied, then the “Case-buffers” and the “Classifier-engine” are dynamically updated.

5 Results and Discussion

As shown in Figs. 3, 4 and in Table 2, we can see that although the learning mechanism uses only small data increments, it succeeds to perform good and consistent results. Figures 3 and 4 represent the dynamics of group set updating for different threshold levels. The process converges in both data sets for all sensitivity levels.

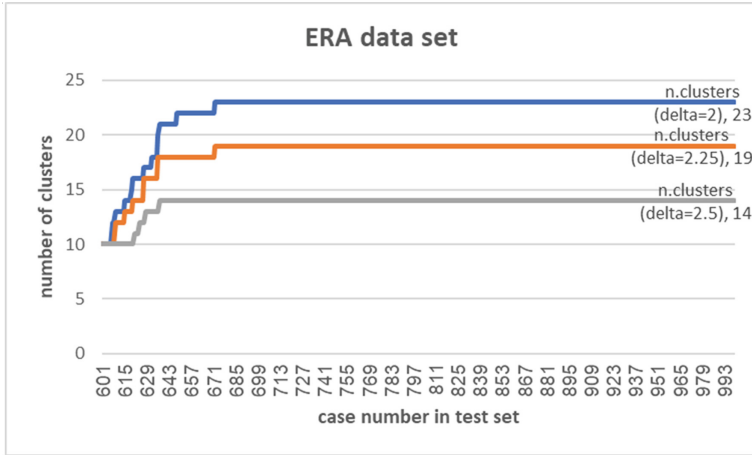


Fig. 3. Rules convergence using k-means with “ERA” dataset.

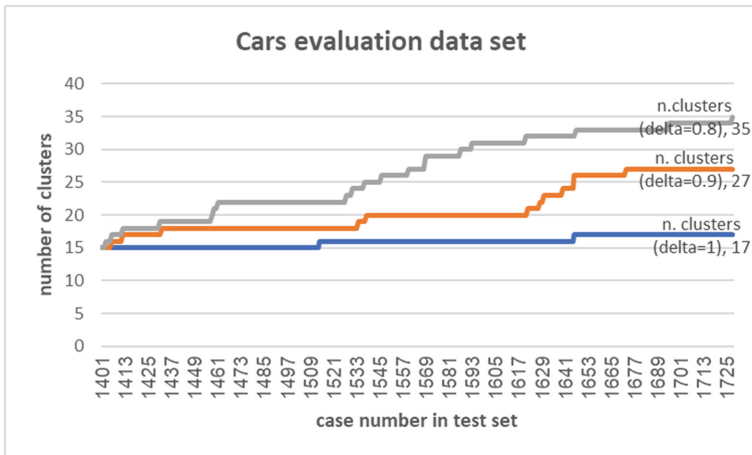


Fig. 4. Rules convergence using k-means with “Car evaluation” dataset.

We can see that in all three threshold values a reach a convergence of the classification process. In order to trace the dependence of aggregate rate of total number of groups on the sensitivity level, we chose three threshold levels for each data set. We can see that the convergence is faster as the threshold refers to lower accuracy value, but even at a high accuracy threshold, a relatively rapid convergence was achieved. The application of this result is very practical: on one hand, the dynamic data environment dictates us to act in real time, that is why we use small increments of data to be able to classify objects immediately; on the other hand, we need to provide good classification results and identify new trends or significant changes in data distribution. The convergence of classification process shows the ability of the proposed model to catch the

critical moments when an update is needed, without too much computational effort. The updated groups set becomes more and more representative, that is why the periods of time between every two updates lengthens.

Table 2 presents the numerical results of all experiments in two data sets. The distance measure RMSE was calculated for each classified item (in most cases the distance between the item and at least one of the existing groups is less than a threshold level, so the item joins the existing group; sometimes the threshold is achieved and the update is needed). The average and standard deviation of all minimal RMSE values are calculated for each experiment. The total number of groups in the end of each experiment is presented in addition. As sensitivity of a threshold level decreases (higher values of δ), the average distance measure grows. This result is expected: if a threshold level is relatively high, less items are defined as “far” or “non-similar” and more items succeed to join existing groups. Their minimal RMSE value is weighted into the calculation of average RMSE and we get bigger result. The same effect usually happens in standard deviation.

Table 2. The dynamic incremental updating of group set, according to threshold level.

Data set	δ	Average RMSE for classified instances	Std. dev.	Number of clusters
ERA Initial number of clusters = 10	2	0.9048	0.6349	23
	2.25	1.1967	0.6677	19
	2.5	1.4455	0.5842	14
Cars evaluation Initial number of clusters = 15	0.8	0.6	0.1133	35
	0.9	0.6871	0.1184	27
	1	0.7433	0.1199	17

In the conclusion of the above facts, we can see that the proposed incremental dynamic mechanism succeeds to achieve good results, that can be adopted in industry or in academical research as well.

6 Conclusions

Dynamic incremental classifier presented in this paper is designed to improve the classification process in state of dynamic data environment. The constant changes in data characteristics and preferences require from the mechanism immediate solutions. In addition to this obligatory condition, the process has to be economic. There is no dispute that the most qualitative solution will be obtained through the update of whole relevant data, but it is not possible in dynamic data environment. We assume that it is not possible to revise all previous data, so we choose to demonstrate the incremental mechanism that functions using small data buffers.

Experiments with different data sets showed that the loss of quality in classification results is not significant and the mechanism succeeds to identify the important changes in data stream and converges during the process.

The further research is planned in different possible directions: dealing with a big data sets that simulate big data environment; new trend and outlier detection; text data processing etc.

Acknowledgment. This work was supported in part by a grant from the MAGNET program of the Israeli Innovation Authority; who also submitted this work as a patent application.

References

1. Fayyad, U., Stolorz, P.: Data mining and KDD: promise and challenges. *Future Gener. Comput. Syst.* **13**, 99–115 (1997). [https://doi.org/10.1016/S0167-739X\(97\)00015-0](https://doi.org/10.1016/S0167-739X(97)00015-0)
2. Gelbard, R., Goldman, O., Spiegel, I.: Investigating diversity of clustering methods: an empirical comparison. *Data Knowl. Eng.* **63**, 155–166 (2007)
3. Fan, J., Han, F., Han, L.: Challenges of big data analysis. *Natl. Sci. Rev.*, 293–314 (2014)
4. Darema, F.: Dynamic data driven applications systems: a new paradigm for application simulations and measurements. In: Bubak, M., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) *ICCS 2004*. LNCS, vol. 3038, pp. 662–669. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24688-6_86
5. Ren, K., Wang, C., Wang, Q.: Security challenges for the public cloud. *IEEE Internet Comput.* **16**, 69–73 (2012). <https://doi.org/10.1109/MIC.2012.14>
6. Li, J., Xu, H.: Suggest what to tag: recommending more precise hashtags based on users' dynamic interests and streaming tweet content. *Knowl.-Based Syst.* **106**, 196–205 (2016). <https://doi.org/10.1016/j.knosys.2016.05.047>
7. Miller, Z., Dickinson, B., Deitrick, W., et al.: Twitter spammer detection using data stream clustering. *Inf. Sci.* **260**, 64–73 (2014). <https://doi.org/10.1016/j.ins.2013.11.016>
8. Siddharth, S., Chauhan, N.C., Bhandery, S.D.: Incremental mining of association rules: a survey. *Int. J. Comput. Sci. Inf. Technol.* **3**, 4041–4074 (2012)
9. Cheung, D.W., Han, J., Ng, V.T., Wong, C.: Maintenance of discovered association rules in large databases: An Incremental Updating Technique, pp. 106–114 (1996)
10. Grira, N., Crucianu, M., Boujemaa, N.: Unsupervised and semi-supervised clustering: a brief survey. In: *Review of Machine Learning Techniques for Processing Multimedia Content* (2005)
11. Sreedhar, G.: *Web Data Mining and the Development of Knowledge-Based Decision Support Systems*. IGI Global, Hershey (2016)
12. Song, Y.C., Meng, H.D., Wang, S.L., et al.: Dynamic and incremental clustering based on density reachable. In: *Fifth International Joint Conference on INC, IMS and IDC, 2009*. NCM 2009, pp. 1307–1310 (2009)
13. Mishra, N., Hsu, M., Dayal, U.: Computer implemented scalable, incremental and parallel clustering based on divide and conquer (2002)
14. Deza, M.M., Deza, E.: *Encyclopedia of Distances*. Springer, Heidelberg (2014). <https://doi.org/10.1007/978-3-662-44342-2>
15. Jain, A.K., Murty, M.N., Flynn, P.L.: Data clustering: a survey. *ACM Comput. Surv.* **31**, 264–323 (1999)
16. Vempala, S., Wang, G., Kannan, R., Cheng, D.: *Techniques for clustering a set of objects* (2010)
17. Thomas, S., Bodagala, S., Alsabti, K., Ranka, S.: An efficient algorithm for the incremental updation of association rules in large databases (1997)

18. Zhang, C-Q., Ou, Y.: Method for data clustering and classification by a graph theory model - network partition into high density subgraphs (2009)
19. Lughofer, E.: A dynamic split-and-merge approach for evolving cluster models. *Evol. Syst.* **3**, 135–151 (2012). <https://doi.org/10.1007/s12530-012-9046-5>
20. Toth, C.K., Grejner-Brzezinska, D.: Extracting dynamic spatial data from airborne imaging sensors to support traffic flow estimation. *ISPRS J. Photogramm. Remote Sens.* **61**, 137–148 (2006). <https://doi.org/10.1016/j.isprsjprs.2006.09.010>
21. Zhang, Y., He, S., Chen, J.: Data gathering optimization by dynamic sensing and routing in rechargeable sensor networks. *IEEE/ACM Trans. Netw.* **24**, 1632–1646 (2016). <https://doi.org/10.1109/TNET.2015.2425146>
22. Okeyo, G., Chen, L., Wang, H., Sterritt, R.: Dynamic sensor data segmentation for real-time knowledge-driven activity recognition. *Pervasive Mob. Comput.* **10**, 155–172 (2014). <https://doi.org/10.1016/j.pmcj.2012.11.004>
23. Ni, Q., Patterson, T., Cleland, I., Nugent, C.: Dynamic detection of window starting positions and its implementation within an activity recognition framework. *J. Biomed. Inform.* **62**, 171–180 (2016). <https://doi.org/10.1016/j.jbi.2016.07.005>
24. Download Python. In: Python.org. <https://www.python.org/downloads/>. Accessed 20 Apr 2018
25. Ben-David, A.: Automatic generation of symbolic multiattribute ordinal knowledge - based DSS's: methodology and applications. *Decis. Sci.* **23**, 1357–1372 (1992)
26. UCI Machine Learning Repository: Data Sets. <https://archive.ics.uci.edu/ml/datasets.html>. Accessed 10 Dec 2016
27. Bohanec, M., Rajkovic, V.: Knowledge acquisition and explanation for multi-attribute decision making, pp. 1–19 (1988)
28. Bittmann, R.M., Gelbard, R.: Visualization of multi-algorithm clustering for better economic decisions—the case of car pricing. *Decis. Support Syst.* **47**, 42–50 (2009). <https://doi.org/10.1016/j.dss.2008.12.012>



Speeding Up Continuous kNN Join by Binary Sketches

Filip Nalepa^(✉), Michal Batko, and Pavel Zezula

Faculty of Informatics, Masaryk University, Brno, Czech Republic
f.nalepa@gmail.com

Abstract. Real-time recommendation is a necessary component of current social applications. It is responsible for suggesting relevant newly published data to the users based on their preferences. By representing the users and the published data in a metric space, each user can be recommended with their k nearest neighbors among the published data, i.e., the kNN join is computed. In this work, we aim at a frequent requirement that only the recently published data are subject of the recommendation, thus a sliding time window is defined and only the data published within the limits of the window can be recommended. Due to large amounts of both the users and the published data, it becomes a challenging task to continuously update the results of the kNN join as new data come into and go out of the sliding window.

We propose a binary sketch-based approximation technique suited especially to cases when the metric distance computation is an expensive operation (e.g., the Euclidean distance in high dimensional vector spaces). It applies cheap Hamming distances to skip over 90% of the expensive metric distance computations. As revealed by our experiments on 4,096 dimensional vectors, the proposed approach significantly outperforms compared existing approaches.

1 Introduction

Social applications have been very popular in recent years, and we can observe a huge amount of data being produced and served to their users. There are a lot of types of such applications specializing at providing their users with different content types, e.g., videos, pictures, music, or news. Due to the huge amount of the produced data, a particular user is not able to get acknowledged with all the data. Instead, there should be appropriate filtering functionalities to deliver to the users only data of their highest interest. A real-time recommendation system is often a necessary part of such an application which informs the users about new published content. One of common techniques of the recommendation is the content-based filtering when the published data are suggested to a user based on the data content and on the user's preferences.

The metric space approach [14] has frequently been used to represent the content of published data and preferences of users. A distance function defined for a given metric space expresses similarity between two data items of the space.

The recommendation is based on evaluating kNN (k-nearest-neighbors) queries for individual users. That means a given user is recommended with k published data items which are the most similar to the user's preferences according to the distance function. In practice, so called feature vectors are frequently used to represent the published data content and the user preferences, and the distance function is the Euclidean distance in a vector space.

One typical scenario is that only recently published data are a subject of the recommendation, i.e., a sliding time window is defined and only the data published within the limits of the window can be recommended (illustrated in Fig. 1). To perform the recommendation in real time, existing approaches maintain the current recommended items for each user, and the results are updated only when a new data item appears in the sliding window or when an old data item expires as it disappears from the window. The challenge is to update the recommendation results efficiently when there are a lot of users (e.g., hundreds of thousands) and there are a lot of published data in the time window (e.g., hundreds of thousands).

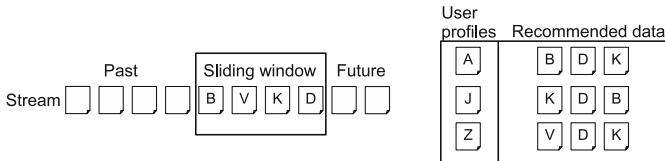


Fig. 1. Recommendation system

Recently, feature vectors of thousands of dimensions have been used to effectively represent the data contents, for instance, Caffe image descriptors [6] are 4,096 dimensional vectors. However, existing approaches for real-time kNN-based recommendation were not designed to handle such high-dimensional spaces, and according to our experiments, their efficiency on these data is poor.

In this paper, we propose an approximation technique for real-time kNN recommendation, which can efficiently process such high-dimensional vector data. The proposed binary sketch based technique applies cheap Hamming distances to avoid expensive metric distance computations. Since it is an approximation method, it can occur that some published data which should be recommended to a user are missed. However, parameters of the proposed method enable to set a suitable trade-off between the quality of the recommendation results and the processing speed. In the next section, we provide a formal definition of the problem as the kNN similarity join; Sect. 3 describes its generic solution. Existing specific solutions are presented in Sect. 4. Our approach is thoroughly explained in Sect. 5. In Sect. 6, the self join variant of the problem is studied. Results of experimental evaluation are presented in Sect. 7.

2 Problem Definition

This section contains a formal definition of the problem solved in this work. Similarity can be universally modeled using metric space (D, d) abstraction [14], where D is a domain of arbitrary objects and d is a total distance function $d : D \times D \rightarrow R$. The distance between any two objects from D corresponds to the level of their dissimilarity ($d(p, p) = 0, d(o, p) \geq 0$).

Let $Q = \{q_1, q_2, \dots, q_n\} \subseteq D$ be a set of query objects. Let $S = ((s_1, t_1), (s_2, t_2), \dots)$ be a stream of pairs (s_i, t_i) where $s_i \in D$ is a metric space object and t_i is its creation time such that $t_i \leq t_{i+1}$ for each $i \geq 1$. Let T be a fixed time interval. For each time t , we define the dataset $W_t = \{s_i | t - T \leq t_i \leq t\}$ as a set of stream objects available at the time t (i.e., stream objects in the sliding time window at time t). The task is to be able to efficiently evaluate a kNN join operation at an arbitrary time t . We suppose Q and W_t can fit into the main memory.

The kNN join at the time t returns the k nearest neighbors from W_t for each query object from Q . The formal definition of the k -nearest-neighbors (kNN) query for a given W_t follows:

$$NN(q, W_t, k) = A \subseteq W_t \text{ where } |A| = k \wedge \forall x \in A, y \in W_t - A : d(q, x) \leq d(q, y)$$

supposing $|W_t| \geq k$.

The formal definition of the kNN join is:

$$NNJoin(Q, W_t, k) = \{(q, NN(q, W_t, k)) | q \in Q\}$$

To solve this task, we employ a commonly used approach which maintains the current result of the kNN join and updates it at the time t if and only if $W_t \neq W_{t-1}$. This situation occurs when there exists such (s_i, t_i) in the stream S where $t_i = t$ or $t_i + T + 1 = t$, i.e., when s_i is added to or removed from W_t , respectively.

We focus on minimizing the average time needed to update the kNN join result when a single object is added to or removed from W_t .

3 General Approach

As mentioned in the previous section, a general approach to the continuous kNN join is to keep the current result of a kNN query for each query object in Q .

When a new stream object s_i arrives (i.e., it becomes an element of W_t at the current time t), the set of affected query objects is identified, i.e., such query objects for which s_i should be among their k nearest neighbors. This operation is known as a reverse kNN query:

$$RNN(s_i, Q, W_t, k) = A \subseteq Q \text{ where } \forall q \in A : s_i \in NN(q, W_t, k) \\ \wedge \forall q \in Q - A : s_i \notin NN(q, W_t, k)$$

The kNN query results are updated for each affected query object q_j such that the new stream object s_i is added to k nearest neighbors of q_j and its $(k + 1)^{\text{th}}$ nearest neighbor is removed from the result.

When a stream object s_i expires at the time t (i.e., $t_i + T + 1 = t$), we identify such query objects for which s_i is among their k nearest neighbors. For each affected query object q_j , s_i is removed from the kNN result and a kNN query is issued to find a stream object from W_t which becomes a new member of the k nearest neighbors of q_j .

Considering a naive sequential approach, the metric distance has to be computed between a given stream object and every query object during a reverse kNN query, and between a given query object and every stream object in a corresponding W_t to evaluate a kNN query. To process reverse kNN and kNN queries efficiently, it is desired to keep the number of distance computations at low levels since it is often an expensive operation (for example, the Euclidean distance at high-dimensional vector spaces).

4 Related Work

In this section, we summarize published works providing efficient solutions for the continuous kNN join.

Authors of [1] propose a grid-based approach for exact solution of the continuous kNN join. They maintain queues of stream objects which can possibly get among the k nearest neighbors of individual query objects. However, their approach was designed for low-dimensional vector spaces (studied on 2 and 3 dimensions).

In [13], the authors present the Sphere-tree for efficient evaluation of reverse kNN queries in the context of the kNN join. The Sphere-tree is based on the concept of the M-tree [4] where the leaf nodes keep distances of individual query objects to their current k^{th} nearest neighbor. When a stream object is removed, they use the iDistance tree structure [12] to efficiently search for new nearest neighbors of the affected query objects. The work considers also the self kNN join, i.e., the set of stream objects is used also as the set of query objects. However, the approach is not suited to high-dimensional vector spaces as was shown in [11] on 128-dimensional data.

Another approach to the continuous kNN join is described in [11]. The paper focuses just on the phase when a new stream object arrives and affected query objects have to be found. For this purpose, the authors designed the HDR-tree based on the principal component analysis (PCA) dimensionality reduction. There is also presented the HDR*-tree based on the random projection dimensionality reduction. The HDR*-tree is capable of approximate search which brings increased efficiency at the cost of reduced result quality. We compare to this approach in Sect. 7.8.

Snapshot kNN joins are the topic of [5], where the problem of returning results of the kNN joins at distinct time snapshots is tackled. Stream objects which arrived since the previous snapshot are processed in batch fashion. The

authors consider the scenario when there is no time window and the stream objects do not expire, and they propose a solution based on the MapReduce paradigm. This solution is intended for processing large volume of data which requires distributed computational environment.

A slightly different problem is dealt with in [9]. The authors study the continuous self join when the join condition is defined as the maximum metric distance between two objects. Given a stream of timestamped vectors, the task is to identify pairs of similar vectors, i.e., such vectors whose distance is below a given threshold. In principle, such a task requires an unbounded memory since no data item can be forgotten as it may be similar to a data item that appears in the future. To overcome this, a time-dependent distance function is defined which decreases exponentially with the difference in the arrival times of the data items. The authors propose an indexing technique to solve such a self join task efficiently.

5 Binary Sketch Approach

This section presents the binary sketch-based approximation technique which we propose for efficient processing of the reverse kNN queries and kNN queries.

We start with the definition of a binary sketch [7]. Consider a set of pivot pairs: $P = \{(p_{11}, p_{12}), (p_{21}, p_{22}), \dots, (p_{m1}, p_{m2})\}$ where $p_{ij} \in D$. The sketch of an object $o \in D$ is a tuple (b_1, b_2, \dots, b_m) where

- $b_i = 0$ iff $d(o, p_{i1}) < d(o, p_{i2})$
- $b_i = 1$ otherwise

Using binary sketches, we approximate a position of an object in the metric space by comparing its distance to the fixed objects (pivots). Having computed sketches of two objects, we can estimate their metric distance by computing the Hamming distance¹ of the sketches and converting the Hamming distance to the original metric distance (inspired by [8]). Since the computation of the Hamming distance is likely to be much faster than the computation of an expensive metric distance, this can speed up query evaluations.

5.1 Hamming Distance vs. Metric Distance

To be able to convert between the Hamming and the metric distances, we use a pre-trained nondecreasing distance conversion function which maps a metric distance md to a Hamming distance hd : $distConv(md) = hd$.

To train the conversion function, we compute both the Hamming distances and the metric distances between pairs of training metric objects. Subsequently, we assign the minimum Hamming distance hd to a given metric distance md so that the following holds. Given two objects in the metric space with their

¹ A Hamming distance of two sketches is computed as the number of positions in which their values differ.

metric distance md , the probability that their Hamming distance is at most hd has to be at least $distProb$, which is a parameter of the training (illustrated in Fig. 2). By altering this parameter, it is possible to control the trade-off between the efficiency and result quality of the continuous kNN join as will be explained later.

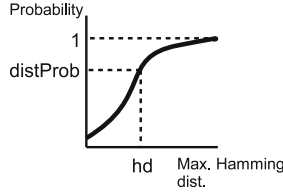


Fig. 2. $distConv$ computation for a given md – the curve expresses the probability that two objects with the metric distance md are within a corresponding hamming distance

To improve the precision of the distance conversion, we can use the fact that the set of query objects Q is known ahead. The conversion function can be defined for each query object separately by considering only the training distances to the given query object. In Sect. 7.4, we compare these two training approaches: global conversion function and multiple conversion functions.

5.2 Pivot Selection

An important aspect of our approach is appropriate selection of the pivot pairs. We rely on the properties stated in [7] to select the pivot pairs: (1) bit values of each sketch should strongly depend on the position of the corresponding object in the metric space; (2) the sketches should have balanced bits, i.e., each bit should be set to 1 in one half of the sketches, and (3) bits of sketches should be mutually as low correlated as possible.

5.3 Sketch Filtering

In the following part, we explain how the sketches are used to speed up evaluation of reverse kNN queries and kNN queries.

For each query object from Q , we keep its sketch and its k nearest neighbors from the current W_t . In addition, we keep the metric distance to its k^{th} nearest neighbor denoted as $metricDistLimit$. We also convert $metricDistLimit$ to $hamDistLimit$ using the conversion function: $distConv(metricDistLimit) = hamDistLimit$.

The pseudocode in Algorithm 1 shows how an arrival of a new stream object is handled. When a new stream object s arrives, its sketch is computed. The next step is to search for the query objects which should take s into their k -nearest-neighbors groups. For each query object $q \in Q$, the Hamming distance

hd between q and s is computed. If $hd \leq hamDistLimit$, we have to compute also the metric distance between s and q to find out whether they are closer than the current k^{th} nearest neighbor of q , i.e., if $d(q, s) < metricDistLimit$. If this is true, the nearest neighbors of q are updated including the update of the $metricDistLimit$ and $hamDistLimit$ values. Otherwise if $hd > hamDistLimit$, the computation of the metric distance $d(q, s)$ can be skipped since it is, with high probability, greater than $metricDistLimit$. By lowering the $distProb$ parameter in the training phase (Sect. 5.1), the number of skipped metric distance computations is increased. On the other hand, this also decreases the quality of the join results since some of the query objects can be skipped even though their k -nearest-neighbors result should be updated.

Algorithm 1. Reverse kNN algorithm

```

function onStreamObjectArrival( $s$ )
   $s.sketch \leftarrow computeSketch(s)$ 
  for all  $q \in Q$  do
    if  $hamDist(q.sketch, s.sketch) \leq q.hamDistLimit$  then
      if  $metricDist(q, s) < q.metricDistLimit$  then
         $q.updateNearestNeighbors(s)$ 

```

Algorithm 2 is run when an object in the stream expires, i.e., it leaves the sliding window. In such a case, we find a new k^{th} nearest neighbor for every affected query object q , i.e., such query objects which had s among their k nearest neighbors. Instead of computing the metric distance between q and every stream object in the current W_t , we apply the sketch filtering to skip some of the computations: Let a be the k^{th} nearest neighbor of q just before s is removed. Let b be the new k^{th} nearest neighbor of q which we want to find after the s 's removal. Assuming the distribution of the metric space data currently in the sliding window does not change a lot when s is removed, b is likely to have a similar metric distance to q as a to q , i.e., $d(q, b) = d(q, a) + \delta$ where δ is a small nonnegative number. This allows us to apply the q 's current value of $hamDistLimit$ to filter out inadequate stream objects by computing the Hamming distances between q and the stream objects in W_t . The metric distance is computed only for those stream objects which pass through this Hamming distance limit. In rare cases, no stream object passes through the sketch filtering. If such a situation occurs, we can fall back to the naive solution and compute the metric distance to all the stream objects to find the new neighbor b .

5.4 Cost Analysis

In this section, we estimate the cost of updating the kNN join result when a stream object enters or is removed from the sliding window.

When a new stream object enters the sliding window, the reverse kNN query is performed.

$$cost_{rkNN} = \mathcal{O}(2|P| \cdot cost_{metricDist} + |Q| \cdot cost_{hamDist} + \alpha \cdot cost_{metricDist})$$

Algorithm 2. New k^{th} neighbors algorithm

```

function onStreamObjectRemoval( $s$ )
  for all  $q \in \text{affectedQueryObjects}(s)$  do
    removeNeighbor( $q, s$ )
     $\text{newNeighborFound} \leftarrow \text{false}$ 
    for all  $so \in W_{\text{now}()}$  do
      if !isAmongNeighbors( $q, so$ ) and  $\text{hamDist}(q.\text{sketch}, so.\text{sketch}) \leq q.\text{hamDistLimit}$ 
then
      if ! $\text{newNeighborFound}$  or  $\text{metricDist}(q, so) < q.\text{metricDistLimit}$  then
         $\text{newNeighborFound} \leftarrow \text{true}$ 
         $q.\text{updateNearestNeighbors}(so)$ 
      if ! $\text{newNeighborFound}$  then
        runNaiveKNNQuery( $q$ )

```

where $|P|$ is the number of pivot pairs used for generating the sketches; $\text{cost}_{\text{metrDist}}$ is the cost to evaluate a single metric distance; $\text{cost}_{\text{hamDist}}$ is the cost to evaluate a single Hamming distance; α is the number of query objects which pass through the sketch filter. We suppose $\text{cost}_{\text{metrDist}} \gg \text{cost}_{\text{hamDist}}$. The value of α is proportional to the probability parameter distProb of the training phase for the distConv function. In addition, α is proportional to the number of truly affected query objects which is estimated as $\frac{k|Q|}{|W_t|}$ [11].

When a stream object leaves the sliding window, a new k^{th} nearest neighbor is found for each affected query object. The estimate is $\frac{k|Q|}{|W_t|}$ affected query objects [11]. The cost of such a single kNN operation can be expressed as follows:

$$\text{cost}_{kNN} = \mathcal{O}((|W_t| - k + 1) \cdot \text{cost}_{\text{hamDist}} + \beta \cdot \text{cost}_{\text{metrDist}})$$

where β is the number of stream objects which pass through the sketch filter. The value of β is proportional to the probability parameter distProb and to the expression $|W_t| - k + 1$ which represents the number of stream objects in the sliding window excluding the ones which are already among the $k - 1$ nearest neighbors of q . We do not consider the pivot set size for the cost analysis since the sketches were computed on the stream object arrivals.

6 Self Join

Continuous kNN self join is a useful operation in certain situations. For instance, consider a news portal. While a user is reading an item of news, the system can recommend them other recent items of news which are the most similar to this one. The set of query objects is not static anymore in such a case, and it equals the set of the current stream objects, i.e., $Q = W_t$ at the given time t .

The proposed sketch-based approach is applicable also to this scenario. When a new stream object arrives, it additionally becomes a new query object (i.e., it is added to Q), and its k nearest neighbors have to be found. This step can be executed together with the reverse kNN algorithm as it is specified by Algorithm 3.

While searching for affected query objects, we also update the nearest neighbors of the newly arrived object (lines 10 and 11). After all the affected query objects are found, we may already have the k nearest neighbors of the new object s selected from the objects which have been examined so far. It remains to process *remainingObjects* to possibly update the nearest neighbors of s (lines 12–15). Based on the already collected data, we use the values of *metricDistLimit* and *hamDistLimit* to speed up the processing of *remainingObjects*.

Algorithm 3. Self join – stream object arrival

```

1: function onStreamObjectArrival( $s$ )
2:    $s.sketch \leftarrow \text{computeSketch}(s)$ 
3:    $remainingObjects \leftarrow Q$ 
4:   for all  $q \in Q$  do
5:     if  $\text{hamDist}(q.sketch, s.sketch) \leq q.hamDistLimit$  then
6:        $md \leftarrow \text{metricDist}(q, s)$ 
7:       if  $md < q.metricDistLimit$  then
8:          $q.updateNearestNeighbors(s)$ 
9:        $remainingObjects \leftarrow remainingObjects \setminus \{q\}$ 
10:      if  $s.countNeighbors() < k$  or  $s.metricDistLimit > md$  then
11:         $s.updateNearestNeighbors(q)$ 
12:      // continue with kNN query
13:      for all  $q \in remainingObjects$  do
14:        if  $s.countNeighbors() < k$  or  $\text{hamDist}(s.sketch, q.sketch) \leq s.hamDistLimit$  then
15:          if  $\text{metricDist}(s, q) < s.metricDistLimit$  then
16:             $s.updateNearestNeighbors(q)$ 
17:       $Q \leftarrow Q \cup \{s\}$ 

```

When an object expires, we can use the Algorithm 2 to handle the situation. In addition, the expired object is removed from Q .

Since the set of the query objects Q dynamically changes, the conversion function *distConv* cannot be trained for individual query objects, but the global *distConv* has to be used instead as described in Sect. 5.1.

7 Experiments

This section contains experimental evaluation results of the proposed sketch-based technique. We analyze the impact of the individual parameters on the efficiency and the effectiveness of the processing. Namely, we study the *distConv* function in Sect. 7.4; the effect of the number of nearest neighbors k is analyzed in Sect. 7.5; other studied parameters are the size of the sliding window in Sect. 7.6 and the size of the query object set in Sect. 7.7. The comparison of our approach to other existing approaches is provided in Sect. 7.8.

7.1 Datasets

For the experiments, we use the Profimedia dataset of images [2], which is a large-scale dataset for evaluation of content-based image retrieval systems. From

the images, we extracted visual-feature Caffe descriptors [6] (4,096 dimensional vectors) and created three distinct sets of vectors used for different purposes: One is used as a set of query objects Q ; another one is used for representing stream objects; the last one serves as training data for the *distConv* function.

In Sect. 7.8, we use also 128 dimensional wavelet texture data from NUS-WIDE Image Data Set [3], which we also separated into three sets as described above.

7.2 Quality Metrics

Since the proposed solution for the continuous kNN join is an approximate approach, we measure the quality of the results using *recall* and *precision* metrics. We altered their original definition (as in [14]) for the use in our scenario.

For each query object in Q , we measure the number of correctly identified stream objects as their nearest neighbors throughout an experiment and compute the particular metric as specified below.

$$recall = \frac{\sum_{i=1}^n |A_i \cap E_i|}{\sum_{i=1}^n |E_i|}$$

$$precision = \frac{\sum_{i=1}^n |A_i \cap E_i|}{\sum_{i=1}^n |A_i|}$$

where n is the size of Q ; A_i is a set of stream objects appeared in the approximate kNN join results for query object q_i throughout the experiment; E_i is a set of stream objects appeared in the exact kNN join results for query object q_i throughout the experiment.

7.3 Setup

All the tested techniques were implemented using Java language, and the experiments were run on Intel Xeon 2.00 GHz.

One step of each experiment consists of an insertion of a new stream object to the sliding window and a deletion of the oldest stream object from the sliding window. This ensures that the number of stream objects in the sliding window is constant (± 1) during the experiment. We run 1,000 such steps in each experiment, and we measure efficiency related properties, the recall, and the precision.

For each experiment, we set the sliding window size, the number of query objects and the number of neighbors to search for, i.e., k . The default value of k is 10 if not stated otherwise. We compare the efficiency of our approach to the naive approach when the metric distance is computed to all query objects on a reversed kNN query, and to all stream objects on a kNN query.

Before each experiment, we load initial stream objects according to the defined window size, and we load initial join results for the individual query objects. We do not include these initial join results into the computation of the recall and the precision metrics.

7.4 Distance Conversion Function

In the first set of experiments, we compare strategies of converting between the Hamming and the metric distances as described in Sect. 5.1. Specifically, we compare the situations when there is a global conversion function and when there are multiple conversion functions (one for every query object). In addition, we test different settings of the probability parameter *distProb* and different numbers of pivot pairs.

The window size in each experiment is 200,000, and there are 100,000 query objects in Q . The value of *distProb* ranges from 0.1 to 0.75. The processing times are presented in Fig. 3. Each bar is divided into two parts representing the time spent with insertions of new stream objects into the sliding window and deletions of the oldest stream objects. Overall, nearly 5,000 query objects were affected by the stream object deletions, which corresponds to the estimate in Sect. 5.4. Additionally, since the sliding window is twice the size as the query object set, the overall deletion time outweighs the insertion time. We measured also numbers of the metric distance computations during each experiment, and they strongly correlate with the processing times.

By decreasing the number of pivots from 512 to 128, we observe an increased number of the metric distance computations (and higher processing times) since more objects pass through the sketch filter. Lowering the *distProb* parameter of the *distConv* function causes less objects to pass the sketch filter, which lowers the number of distance computations. We can also see a substantial difference between the global and the individual *distConv* functions. According to these results, we stick to 512 pivots and multiple *distConv* functions in following experiments.

For comparison, the naive approach (not in the graphs) ran 5,912 s.

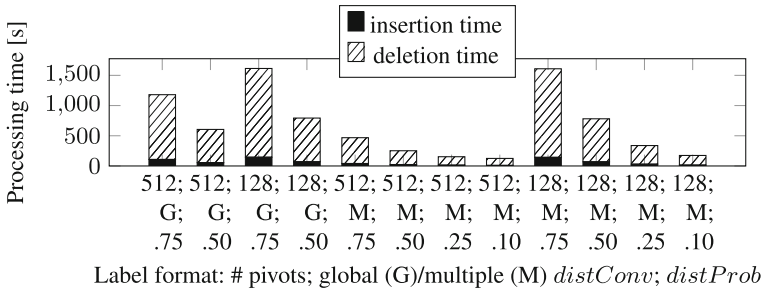


Fig. 3. Distance conversion function experiments – processing times; $|W_t| = 200,000$; $|Q| = 100,000$

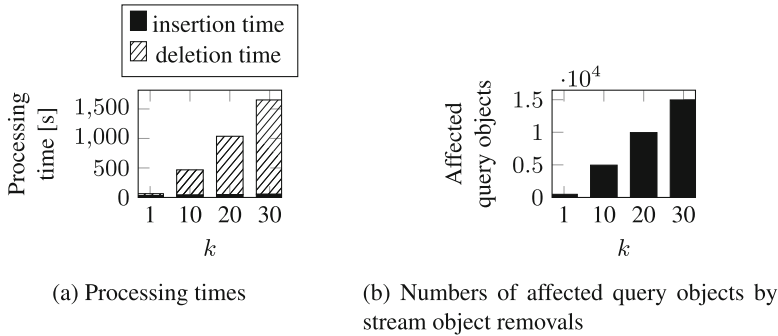
Table 1 captures the quality of the continuous kNN join results. By lowering the *distProb* parameter, the result quality reduces. We fix the *distProb* parameter to 0.75 in further experiments since it provides almost precise results.

Table 1. Recall and precision; $|W_t| = 200,000$; $|Q| = 100,000$

<i>convDist</i>	Global				Multiple							
# pivots	512		128		512				128			
<i>distProb</i>	0.75	0.50	0.75	0.50	0.75	0.50	0.25	0.10	0.75	0.50	0.25	0.10
recall	0.99	0.95	0.97	0.90	≈ 1	0.98	0.91	0.78	≈ 1	0.98	0.91	0.76
precision	0.99	0.96	0.98	0.93	≈ 1	0.99	0.94	0.84	≈ 1	0.98	0.94	0.83

7.5 k Effect

Figure 4a shows the processing times for various values of k . The window size was set to 200,000, and we used 100,000 query objects. The differences in the times are caused by the numbers of affected query objects when an old stream object leaves the sliding window (see Sect. 5.4 for details). A new k^{th} nearest neighbor must be found for each such affected query object. The numbers of the affected query objects by stream object removals are shown in Fig. 4b. The naive approach took 1,063 s, 5,912 s, 11,272 s and 16,614 s for $k = 1, 10, 20,$ and 30 objects, respectively.

**Fig. 4.** k effect; $|W_t| = 200,000$; $|Q| = 100,000$

7.6 Window Size Effect

The next set of experiments compares scenarios with different sizes of the sliding window. We used 100,000 query objects. The results are captured by Fig. 5. Interestingly, the processing time decreases by increasing the window size. This is caused by the numbers of affected query objects when an old stream object leaves the sliding window. Enlarging the window size lowers the number of affected query objects by a stream object removal. The processing time does not drop linearly since the cost to find a new k^{th} nearest neighbor is more expensive for larger windows. (See Sect. 5.4 for details). To compare with the naive approach, the processing times were 5,891 s, 4,957 s and 5,859 s for the window sizes 100,000, 200,000, and 300,000, respectively.

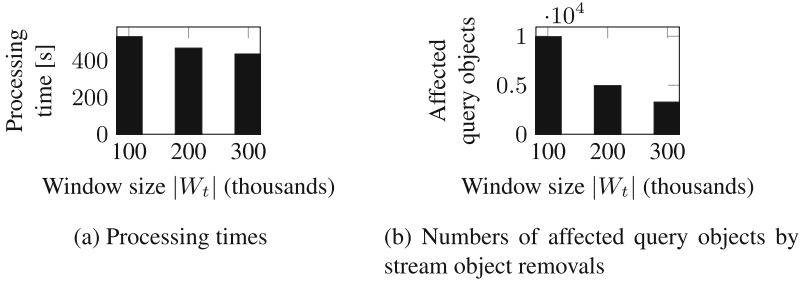


Fig. 5. Window size effect; $|Q| = 100,000$

7.7 Query Object Set Size Effect

Another set of experiments is devoted to the comparison of different sizes of query object sets. The window size was 200,000. Figure 6 displays the results. The processing times increase with an increasing size of Q due to the raise of the numbers of affected query objects (details in Sect. 5.4). The processing times for the naive approach were 2,904 s, 5,912 s and 11,984 s for the Q size of 50,000, 100,000, and 200,000, respectively.

7.8 Comparison to Existing Techniques

This section contains comparison of the proposed sketch-based approach to other existing techniques. We compare to the HDR*-tree [11] designed specifically for approximate solutions of the continuous kNN join. The HDR*-tree aims only at the phase when a stream object arrives at the sliding window and a reversed kNN query is executed. We used the M-Index [10] for the other situation when a stream object leaves the sliding window and new k^{th} nearest neighbors have to be found. The M-Index is a state-of-the-art indexing structure capable of evaluating approximate kNN queries amongst other operations.

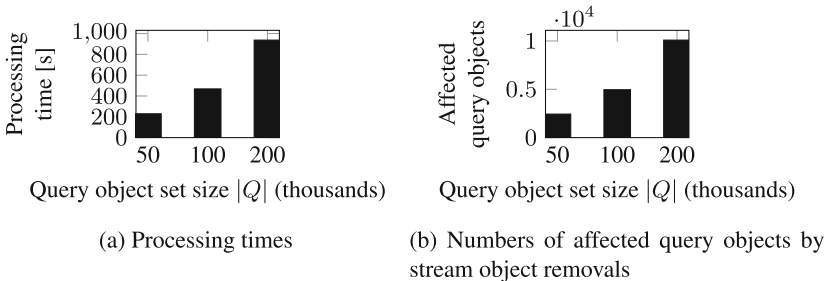


Fig. 6. Query object set size effect; $|W_t| = 200,000$

We used publicly accessible implementation of the HDR*-tree². Since the HDR*-tree was originally tested on 128 dimensional vectors from NUS-WIDE Image Data Set [3], we included also this dataset to the comparison.

Let us start with the NUS-WIDE dataset. We used the query object set Q of size 50,000, the size of the sliding window was fixed to 200,000, and the $distProb$ was set to 0.5. Figure 7a shows the total insertion times needed by the approaches using the HDR*-tree and the sketches. The sketch-based approach managed to be 10% faster. Figure 7b depicts the total time spent by handling removals from the sliding window; M-Index was 20% faster than the sketch based approach. Figures 7c and d show the numbers of metric distance computations.

In the used implementation, the average times to compute a Hamming distance and a metric distance were $6 \cdot 10^{-5}$ ms and $4.2 \cdot 10^{-4}$ ms respectively, which makes the Hamming distance computation only about 7 times faster than the metric distance computation. As there were 57 times more Hamming distance computations than the metric distance ones during the experiment, too much time was spent by the Hamming distance computations as can be seen in the graphs.

The approximation levels of HDR*-tree and M-Index were configured to match the approximation of the sketch-based technique. The recall and the precision for the HDR*-tree+M-Index approach were 0.98 and ≈ 1 respectively. For the sketch-based approach, the recall and the precision were 0.99 and ≈ 1 respectively. To compare with the naive approach, the insertion time was 31 s; the deletion time was 162 s. In conclusion, the M-Index (handling deletions) and the sketch-based approach (handling insertions) can be combined to achieve the best results on this dataset.

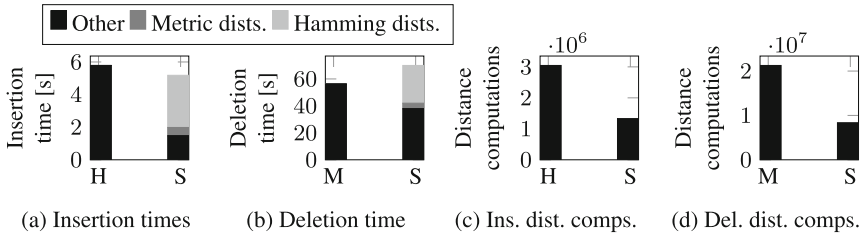


Fig. 7. Comparing approaches: 128 dimensional vectors; $|Q| = 50,000$; $|W_i| = 200,000$ Notation: H: HDR*-tree; M: M-Index; S: Sketches (The legend applies to the sketch-based approach only.)

Let us move back to the Caffe descriptors. The set of query objects Q was set to 50,000; the window size was 200,000. The approximation level of M-Index was configured so that its processing time matches the sketch-based approach. Figure 8a and b capture the total insertion and the deletion times of the studied

² <https://github.com/chongzi1990/continuousNNjoin.git>.

approaches. Although the HDR*-tree managed to skip 24% of the original metric distance computations, there was a significant overhead at intermediate levels of the tree.

The average times to compute a Hamming distance and a metric distance were $4 \cdot 10^{-5}$ ms and $5.4 \cdot 10^{-3}$ ms respectively, making the Hamming distance computations almost 140 times faster than the metric ones considering the used implementation. The sketch-based approach takes advantage of this difference to speed up the processing.

As it turns out, the proposed approach performs better on such high-dimensional data than the compared techniques regarding both the processing times and the quality of the results. The recall and the precision for the HDR*-tree+M-Index approach were 0.90 and 0.59 respectively. For the sketch-based approach, the recall and the precision were ≈ 1 .

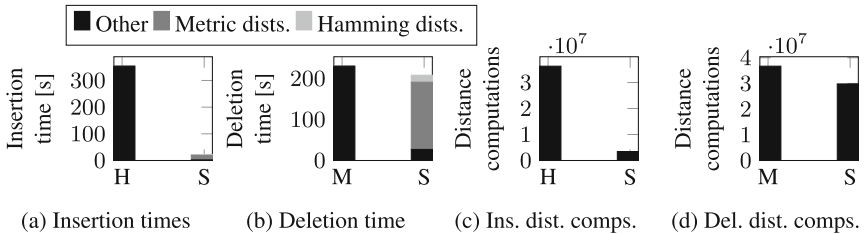


Fig. 8. Comparing approaches: 4,096 dimensional vectors; $|Q| = 50,000$; $|W_i| = 200,000$; notation: H: HDR*-tree; M: M-Index; S: Sketches (The legend applies to the sketch-based approach only.)

8 Conclusion

We have presented approximation sketch-based technique that efficiently solves the continuous kNN join problem in generic metric space data domains. The approach works with cheap Hamming distances to skip over 90% of expensive distance computations compared to the naive solution. The technique is used to handle both reverse kNN and kNN queries as opposed to existing approaches, which mostly do not provide an efficient universal technique for both of them. The proposed approach is suited also for the self join scenario. Although it is an approximation technique, we can obtain nearly precise results while still achieving good efficiency. The technique was compared to the HDR*-tree, that was proposed for evaluating reverse kNN queries, and to the M-Index used for kNN queries. The sketch-based approach achieved significantly better processing times and quality of the results based on experiments with high dimensional vectors (4,096 dimensions).

Acknowledgements. This work was supported by the Czech national research project GA16-18889S.

References

1. Böhm, C., Ooi, B.C., Plant, C., Yan, Y.: Efficiently processing continuous k-nn queries on data streams. In: ICDE, pp. 156–165. IEEE Computer Society (2007)
2. Budikova, P., Batko, M., Zezula, P.: Evaluation platform for content-based image retrieval systems. In: Gradmann, S., Borri, F., Meghini, C., Schuldt, H. (eds.) TPD 2011. LNCS, vol. 6966, pp. 130–142. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24469-8_15
3. Chua, T., Tang, J., Hong, R., Li, H., Luo, Z., Zheng, Y.: NUS-WIDE: a real-world web image database from national university of singapore. In: CIVR, ACM (2009)
4. Ciaccia, P., Patella, M., Zezula, P.: M-tree: an efficient access method for similarity search in metric spaces. In: VLDB, pp. 426–435. Morgan Kaufmann (1997)
5. Hu, Y., Yang, C., Ji, C., Xu, Y., Li, X.: Efficient snapshot KNN join processing for large data using mapreduce. In: ICPADS, pp. 713–720. IEEE Computer Society (2016)
6. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R.B., Guadarrama, S., Darrell, T.: Caffe: convolutional architecture for fast feature embedding. In: ACM Multimedia, pp. 675–678. ACM (2014)
7. Mic, V., Novak, D., Zezula, P.: Improving sketches for similarity search. In: Proceedings of MEMICS, pp. 45–57 (2015)
8. Mic, V., Novak, D., Zezula, P.: Speeding up similarity search by sketches. In: Amsaleg, L., Houle, M.E., Schubert, E. (eds.) SISAP 2016. LNCS, vol. 9939, pp. 250–258. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-319-46759-7_19
9. Morales, G.D.F., Gionis, A.: Streaming similarity self-join. PVLDB **9**(10), 792–803 (2016)
10. Novak, D., Batko, M., Zezula, P.: Metric index: an efficient and scalable solution for precise and approximate similarity search. Inf. Syst. **36**(4), 721–733 (2011)
11. Yang, C., Yu, X., Liu, Y.: Continuous KNN join processing for real-time recommendation. In: ICDM, pp. 640–649. IEEE Computer Society (2014)
12. Yu, C., Ooi, B.C., Tan, K., Jagadish, H.V.: Indexing the distance: an efficient method to KNN processing. In: VLDB, pp. 421–430. Morgan Kaufmann (2001)
13. Yu, C., Zhang, R., Huang, Y., Xiong, H.: High-dimensional knn joins with incremental updates. GeoInformatica **14**(1), 55–82 (2010)
14. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search - The Metric Space Approach, Advances in Database Systems, vol. 32. Kluwer (2006)



Mining Cross-Level Closed Sequential Patterns

Rutba Aman^(✉) and Chowdhury Farhan Ahmed

Department of Computer Science and Engineering,
University of Dhaka, Dhaka, Bangladesh
rutba.aman@gmail.com, farhan@du.ac.bd

Abstract. Multilevel, cross-level and sequential knowledge plays a significant role in our several real-life aspects including market basket analysis, bioinformatics, texts mining etc. Many researchers have proposed various approaches for mining hierarchical patterns. However, some of the existing approaches generate many multilevel and cross-level frequent patterns that fail to fetch quality information. It is extremely difficult to extract any meaningful information from these large number of redundant patterns. There exist some approaches that mines multilevel and cross-level closed patterns but unfortunately, there is no cross-level closed pattern mining method proposed yet which maintain the sequence of itemsets. In this paper, we develop an algorithm, called CCSP (Cross-level Closed Sequential Pattern mining) to conduct cross-level hierarchical patterns that provide maximal information. Our work has made contributions in mining patterns, which express the mixed relationship between the generalized and specialized view of the transaction itemsets. We have extensively evaluated our proposed algorithm's efficiency using a variety of real-life datasets and performing a large number of experiments.

Keywords: Data mining · Sequential pattern · Closed-pattern
Cross-level closed pattern

1 Introduction

Among various promising fields of data mining, pattern mining has become more essential and discussed task in our real life. It consists of discovering interesting, useful, and unexpected patterns in databases. Various types of patterns can be discovered in databases such as sequential patterns [1, 12, 17], closed-frequent patterns [11, 22], closed sequential pattern [4, 21], multilevel pattern [5, 7, 8, 15, 20], cross-level pattern [10, 18], cross-level closed pattern [9] etc.

In many domains, the ordering of events is important. For example, to analyze texts, it is often relevant to consider the order of words in sentences [13]. In network intrusion detection, the order of events is also important [14]. If the sequence of events is not taken into account, this may result in the failure to

discover important patterns. Moreover many fields also support hierarchies that represent the relationships amongst many different concept levels. For example, to discover links between certain genes and diseases, health care companies may use sequential patterns which also have cross/inter level relationship. Weather companies can mine data to discover weather patterns that may help predict future meteorologic events. Online services might discover popular topics among different groups of people and thus provide targeted content and advertisements to their users based on these information.

In today's competitive business world, analysis of the purchase-sale characteristics of transactional database items have become an important issue because sale or purchase of one item may increase/decrease the sale/purchase of other items. For example, in a grocery store there are huge collection of food items. The Table 1 contains a sample transactional database of a store. In order to increase the sale of food items the manager should adopt a strategy to place the items in a way that will impact a customer to buy more items.

Table 1. Transaction database

Tid	Items
t1	Mango, Energy biscuit, Potato chips, Apple juice, Banana
t2	Marry biscuit, Apple juice, Banana, Mum water
t3	Mango, Potato chips, Mum water, Banana
t4	Marry biscuit, Apple juice, Banana, Mum water, chocolate cake
t5	Mango, Potato chips, Banana, Marry biscuit
t6	Energy biscuit, Potato chips, Apple juice, Banana, Fanta

Suppose we know that 40% of customers who buy Potato chips and chocolate cake, also buy Fanta or Sprite. Similarly, we can say that if customers buy any chips and cakes, there is 60% chance that they will buy any kind of soft drinks. The second statement provides more general information for the shop-manager than the previous statement. Using multilevel pattern mining we can find out frequent patterns such as:

1. {Coccola biscuit, Apple juice, Banana, Mum water},
2. {Marry biscuit, Orange juice, Apple, Fresh water},
3. {Coccola biscuit, Mango juice, Orange, Fresh water},
4. {Coccola biscuit, Pineapple juice, Grape, Fresh water},
5. {Biscuit, Juice, Seasonal fruit, Water},
6. {Dry food, Liquid, Fruit, Liquid}.

Now, suppose the manager want to know the information about the sequence of items as well as specific information about only biscuit item not all items but none of the pattern given above can give this kind of information. For example we need a pattern such as {Coccola biscuit, Juice, Seasonal fruit, Water}. From

this pattern we can discover the sequence of itemset as well as specific information about only biscuit. So manager can find out that most of the customer buy **Coccola biscuit**. In this case, which juice, fruit and water are bought by customer is not his consideration. In recent years, many studies have presented to find out different types of patterns but none of them can solve this problem. Using the previous researches either we can find out the information about the sequence of items or specific information of some items as well as general information of other items but in that case they can not maintain the sequence of items.

In this paper, we present a nice solution that efficiently mines the patterns that will give us information about both the sequence of items and specific information of one or more items as well as general information of other items. For example our solution will easily find out the the pattern such as 75% of customers purchase {coccola biscuit, juice, non-seasonal fruit, water} and 25% of customers purchase {marry biscuit, juice, non-seasonal fruit, water} instead of the six patterns given above. So, **key contributions** of this paper are as follows:

- (1) We have introduced the concept of mining cross-level closed pattern which maintain the sequences of items.
- (2) Designed an efficient algorithm for mining cross-level closed sequential pattern by traversing the level-wise generated closed sequence itemsets.
- (3) Real-life datasets are used to show the efficiency and scalability of our approach.

Rest of the paper is organized as follows. The next section presents background and related works. In Sect. 3, we propose CCSP after describing the complete methodology of our idea and a detailed explanation of our proposed algorithm. Experimental results are shown in Sect. 4 and conclusions are given in Sect. 5.

2 Background Study and Related Work

Sequential pattern mining have many real life impact. It was first proposed by Agrawal and Srikant in [1] which was also developed further by them in Generalized Sequential Patterns (GSP) [17], based on the Apriori property [2]. For improving performance, several algorithm have been proposed for sequential pattern mining ever since. Among them some are quite interesting. For instance, SPADE [24], based on a vertical ID-list format whereas PrefixSpan [12] adopts a horizontal format data set representation. Moreover, SPAM [3] is developed for mining long sequential patterns and adopts a vertical bitmap representation.

CLOSET [11], CHARM [25], CLOSET+ [22] and several recently devised methods [6] have been proposed for mining closed itemset. Most of these algorithms need to maintain the already mined frequent closed patterns in order to do pattern closure checking. For reducing the memory usage and search space for

this checking, CLOSET+, adopt a compact two level hash-indexed result-tree structure to store the already mined frequent closed itemset candidates.

CloSpan [23] was used for mining frequent closed sequences that follows the candidate maintenance-and-test approach. As a result it consumes much memory and leads to a huge search space for pattern closure checking when there are many frequent closed sequences. Later more efficient algorithm BIDE [21] was proposed to avoid the candidate maintenance-and-test paradigm, prunes the search space more deeply and checks the pattern closure in a more efficient way while consuming much less memory.

In order to acquire in-depth knowledge, multi-level rule mining was first proposed by Srikant and Agrawal [16]. Further Han and Fu [7] proposed a reduced support threshold methodology to prune many hierarchical redundant rules. Level-crossing association rules mining was proposed in [19]. Then MMCAR [9] was proposed for mining cross-level closed itemsets. In MMCAR cross-level itemsets have been mined by traversing the level-wise generated closed itemset lattice. These closed itemset lattice were generated using CHARM-L [26]. MMCAR only mines cross-level closed frequent itemsets which do not maintain the sequence of transactions.

2.1 Preliminary Terminologies

In order to construct lattices of cardinality $n + 1$, we must to cross two lattices of cardinality n . Some necessary terms and their definition which were demonstrated in some previous works of other researchers are needed to understand for realizing our proposed strategy effectively. These terms and definition are given below.

Ancestor and Descendant Itemsets: Consider that a tuple containing itemset $\{1-* - *\}$ in lattice L_1 and another tuple containing itemset $\{1-1-*, 1-2-*\}$ in lattice L_2 . The prefix of item $\{1-* - *\}$ is similar to the item $\{1-1-*$ and $\{1-2-*\}$. Hence, $\{1-* - *\} = \text{ANC}\{1 - 1-*, 1 - 2-*\}$ and $\{1-1-*$ and $\{1-2-*\} = \text{DES}\{1-* - *\}$.

Conflicting Itemset and Resolved Conflicting Itemset: A conflicting itemset is composed of the a_i^{th} level items of the tuple T1 and a_k^{th} level items of tuple T2 from the two candidate lattices that cannot co-occur in the resulting Cross-level frequent closed itemset (CFCI) of the newly generated tuple S. Because the conflicting items are mutually related in an ancestor-descendant relationship, they belong to the same branch in the tree-like taxonomical structure of the itemset.

Matching Itemset: Consider the tuple containing itemset $\{1-* - *, 2-1-*, 3-2-*\}$ in lattice $L_{1,2}$ and the tuple containing itemset $\{1-1-1, 1-2-1, 2-1-*, 3-2-1\}$ in lattice $L_{2,3}$ for the database of Table 2. While crossing $L_{1,2} \times L_{2,3}$, the matching itemset is $\{2-1-*\}$.

Distinct Itemset: If the tuple with itemset $\{2-* - *, 1-2-*\}$ of lattice $L_{1,2}$ and tuple with itemset $\{1-2-*, 1-1-1\}$ of lattice $L_{2,3}$ is selected for crossing, then

the distinct itemset will be $\{2-* - *\}$ and $\{1-1-1\}$ for lattices $L_{1,2}$ and $L_{2,3}$ respectively.

3 CCSP: Our Proposed Algorithm

In this section, we introduce the CCSP algorithm by answering the following questions: How do we construct encoded transaction database? How do we compute single item frequency for each level? How do we construct sequential closed itemset lattice for specific level? How do we mine cross-level closed sequential patterns?

3.1 Construct Encoded Transaction Database

To represent each item of every level we use n-digit code for an n-level dataset. Items that belong to the top-most level (nearest to the root node) are labeled by level-1 and in this way the level number is incremented as the depth of items'. As a consequence, leaf nodes are belonged to the highest number.

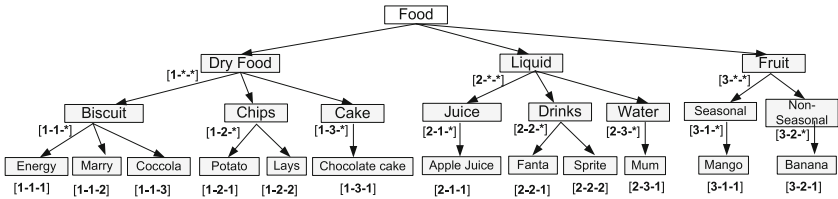


Fig. 1. Hierarchical structure of itemsets

Hence, for the database of Table 1 with 3 levels, we needed 3-digit codes for representing each item. The 1st digit represents the item at the uppermost level-1, the 2nd digit represents the 2nd level (intermediate) item and the last digit represents the leaf item according to the hierarchical tree represented in Fig. 1. For example, the item 1-2-2 belongs to group-1 of level-1, group-2 of level-2 and 2nd item of the leaf level. The transaction database of Table 1 now transforms to the database listed in Table 2.

3.2 Frequency Computation of Items

First, we need to categorize the leaf level items based on their similar properties for working with the data at cross-level and multilevel. Then we extract the single item of each level. The encoding technique described in Sect. 3.1 make this extracting process easy.

To specify the minimum support for every level, the reduced support methodology have been used. According to this process, the user specified minimum

Table 2. Encoded transaction database

Tid	Items
t1	3-1-1, 1-1-1, 1-2-1, 2-1-1, 3-2-1
t2	1-1-2, 2-1-1, 3-2-1, 2-3-1
t3	3-1-1, 1-2-1, 2-3-1, 3-2-1
t4	1-1-2, 2-1-1, 3-2-1, 2-3-1, 1-3-1
t5	3-1-1, 1-2-1, 3-2-1, 1-1-2
t6	1-1-1, 1-2-1, 2-1-1, 3-2-1, 2-2-1

support for any level is usually smaller than the minimum support specified for its higher level. There is always the possibility that the higher level transaction item may occur more frequently than that of the lower level.

As only one scan of the transaction database is required to build the vertical Tidlist for each item of the leaf level, we have used the vertical format of the transaction database. In Figs. 2, 3 and 4 we have shown the vertical format for every level of data that will be used to construct lattices of each specific level.

The Tidlist (a list of transaction Id.) for higher level item can be computed through the union of the lower level items' Tidlists. For example, consider the item hierarchy shown in Fig. 1 and according to Table 2, where Tidlist (1-1-*) = Tidlist (1-1-1) \cup Tidlist (1-1-2) \cup Tidlist (1-1-3) = {1, 6} \cup {2, 4, 5} \cup { \emptyset } = {1, 2, 4, 5, 6}.

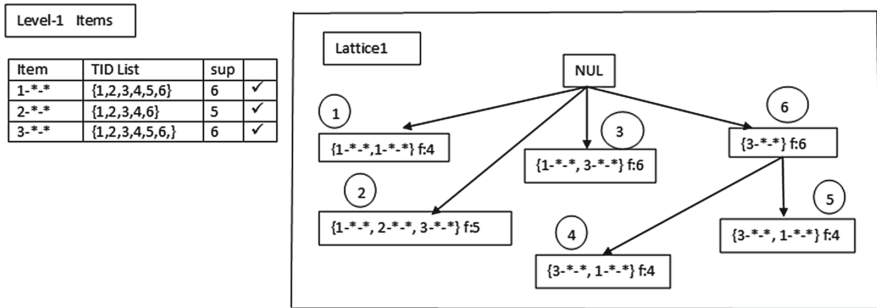


Fig. 2. Vertical format of level-1 itemset tuples and lattice construction

3.3 Specific Level Sequential Closed Itemset Lattice Construction

In this phase, we will generate the sequential closed itemset lattice for each level. At the initialization phase of BIDE [21], we used every frequent single item with their Tidlist, already computed during the scan of the transactional database. BIDE explicitly output the sequential closed itemset lattice for the corresponding level.

Level-2 Items		
Item	TID List	sup
1-1-*	{1,2,4,5,6}	5 ✓
1-2-*	{1,3,5,6}	4 ✓
1-3-*	{4}	1 ✗
2-1-*	{1,2,4,6}	4 ✓
2-3-*	{2,3,4}	3 ✓
3-1-*	{1,3,5}	3 ✓
3-2-*	{1,2,3,4,5,6}	6 ✓

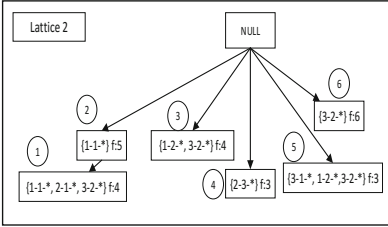


Fig. 3. Vertical format of level-2 itemset tuples and lattice construction

Level-3 Items		
Items	TID List	sup
1-1-1	{1,6}	2 ✓
1-1-2	{2,4,5}	3 ✓
1-2-1	{1,3,5,6}	4 ✓
1-3-1	{4}	1 ✗
2-1-1	{1,2,4,6}	4 ✓
2-3-1	{2,3,4}	3 ✓
3-1-1	{1,3,5}	3 ✓
3-2-1	{1,2,3,4,5,6}	3 ✓

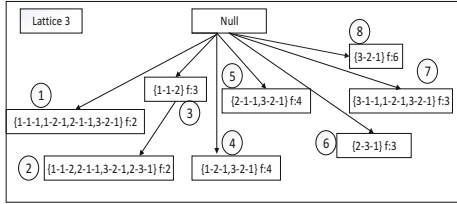


Fig. 4. Vertical format of level-3 itemset tuples and lattice construction

In Fig. 2 we construct the sequential closed itemset lattice using BIDE algorithm for level-1 itemsets. Figures 3 and 4 shows the sequential closed lattice structure for Level-2 and level-3 itemsets respectively.

3.4 Cross-Level Sequential Closed Itemset Mining

Performing a crossing operation among the lattices found previously, using BIDE, we can construct cross-level sequential closed itemset lattice. In general, we know that crossing two lattices from level x ; L_x and level y ; L_y will generate a cross-level lattice, i.e., a lattice of levels x and y ; $L_{x,y}$. For example, crossing two lattices of cardinality 1 will generate a lattice of cardinality 2 and crossing two lattices of cardinality 2 will result in lattices of cardinality 3, and so on.

Again crossing all possible lattices of cardinality n to generate lattices of cardinality $n + 1$ for $n \geq 2$, we need to pruned some search spaces discussed in MMCAR [9]. So we need not to consider crossing all possible pairs of lattices.

By making these lattice structure we can easily find out frequent sequential itemsets. As these are also closed itemsets so mining itemsets are more compact and complete. So it reduces our search space and increase efficiency. Then we need to focus on the determination of the minimum support threshold for a generated lattice by crossing two lattices of different levels with distinct minimum support count thresholds. The approach of determining the minimum support threshold for newly generated cross-level frequent closed itemset lattices is similar to the approach defined in [8]. The minimum support threshold for a lattice $L_{a_1, a_2, \dots, a_{k+1}}$ found by crossing two lattices L_{a_1, a_2, \dots, a_k} with support threshold r_1 and $L_{a_2, a_3, \dots, a_{k+1}}$ with support threshold r_2 is the minimum between r_1 and r_2 .

In Fig. 5, by combining the contributed itemset Segment of L_1 and L_2 we can generate cross-level frequent closed itemset. Here is our another challenge,

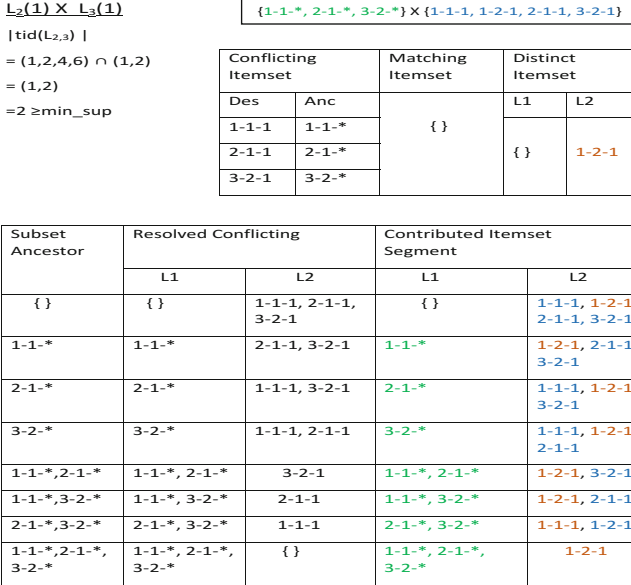


Fig. 5. Crossing 1st element of 2nd and 3rd lattice.

because we also need to maintain the sequence of transaction itemset. One approach is to make all kind of combination of L_1 and L_2 contributed itemsets and then compare the itemsets with main transaction database to find out which one is maintaining the sequence. But it is not an efficient approach. Rather we have proposed an efficient way to generate cross-level sequential closed itemsets from these contributed items segment. When we generate the itemsets from contribute itemsets of lattice L_1 and L_2 , we will follow the sequence of the lattice which is the super-set of the another lattice. For example, in Fig. 5 lattice $L_3(1)$ is the super-set of lattice $L_2(1)$. Now if we maintain the sequence of lattice $L_3(1)$ when generate the itemset for this cross level, our output itemset must be sequential closed for this cross-level. After following this proposal our output itemsets for lattice L_{23} are given below:

- {1-1-*, 1-2-1, 2-1-1, 3-2-1}
- {1-1-1, 1-2-1, 2-1-*, 3-2-1}
- {1-1-1, 1-2-1, 2-1-1, 3-2-*
- {1-1-*, 1-2-1, 2-1-*, 3-2-1}
- {1-1-*, 1-2-1, 2-1-1, 3-2-*
- {1-1-1, 1-2-1, 2-1-*, 3-2-*
- {1-1-*, 1-2-1, 2-1-*, 3-2-1}

In Fig. 6 we can see that support threshold of $L_2(1)$ is 4 and support threshold of $L_3(3)$ is 3. When we cross these two different level lattice, the resultant cross itemset will be frequent if their support will more than or equal to 3. But here we

can see that the resultant cross itemset frequency is 2. So no pattern generated from these cross level will be frequent. Therefore we need not to calculate further from this cross calculation.

$$\begin{aligned}
 & \underline{L_2(1)} \times \underline{L_3(3)} \\
 & \{1-1-*, 2-1-*, 3-2-*\} \times \{1-1-2\} \\
 & |\text{tid}(L_{2,3})| \\
 & = (1,2,4,6) \cap (2,4,6) \\
 & = (2,4) \\
 & = 2 < \text{min_sup}
 \end{aligned}$$

No need to calculate further

Fig. 6. Crossing 1st element of 2nd and 3rd element of 3rd lattice

$$\begin{aligned}
 & \underline{L_{1,2}(1)} \times \underline{L_{2,3}(2)} \\
 & |\text{tid}(L_{1,2,3})| \\
 & = (1,2,4,6) \cap (1,6) \\
 & = (1,6) \\
 & = 2 \geq \text{min_sup}
 \end{aligned}$$

$$= 2 \geq \text{min_sup}$$

No distinct itemset found. So no need to calculate further.

Fig. 7. Crossing 1st element of lattice(1, 2) and 2nd element of lattice(2, 3)

{1-1-*, 2-1-*, 3-2-*} × {1-1-1, 1-2-1, 2-1-*, 3-2-1}				
Conflicting Itemset		Matching Itemset	Distinct Itemset	
			L1	L2
Des	Anc	2-1-*	{ }	{ }
1-1-1	1-*-*			
1-2-1	1-*-*			
3-2-1	3-2-*-*			

Moreover, in Fig. 7, a special case of lattice construction is shown: crossing 1st element of $L_{1,2}$ and 2nd element of $L_{2,3}$. If we cross among these tuples, our finding crossing itemset will as same as it's parent itemset due to the absence of a distinct itemset segment between the tuples. Therefore we need not to calculate further from this cross calculation.

3.5 The CCSP Algorithm

Complete pseudocode for CCSP is shown in Algorithm 1. As input, the Algorithm 1 at line number 1 takes the transaction database, number of levels to construct hierarchy tree and the frequency of that itemset in the database, for each level of items.

Algorithm 1. Cross-level Closed Sequential Pattern(CCSP)

- 1: **procedure** *CCSP*(*SDB*, *level*, *min_sup*)
 - 2: *DI* := *Count_Distinct_Items*(*SDB*)
 - 3: *E_SDB* := *Encode_SequenceDatabase*(*SDB*, *level*, *DI*)
 - 4: **for** *l*:=1 to *level* **do**
 - 5: *FCS_l* := *BIDE*(*E_SDB_l*, *l*, *min_sup*)
 - 6: *L₁*, *L₂* := *User_Input_for_Crossing_among_any_two_levels*
 - 7: *Cross_{L1,L2}* := *Cross_level*(*E_SDB_{L1}*, *E_SDB_{L2}*, *FCS_{L1}*, *FCS_{L2}*)
-

At line 2, the function *Count_Distinct_Items*() counts the total distinct items in the sequence database, defined in Algorithm 2 from line 10 to line 21.

At line 3, this function encoded the items of sequence database according to the corresponding level. Then constructs a closed itemset lattice tuple containing an itemset, an incident transaction list for the itemset, using a data tuple belonging to the set of data tuples of i-th level, and adds the tuple to the corresponding specific level lattice using the BIDE algorithm in each iteration of the for loop, defined in line (4 to 5).

At line 6 of the algorithm CCSP, generate cross-level closed itemset lattices of cardinality $k + 1$ by crossing any two lattices of cardinality k . While crossing two lattices of the form $M_{a_1;a_2;\dots;a_k}$ and $N_{a_2;a_3;\dots;a_{k+1}}$ among the lattices with cardinality k , the algorithm checks for a match between a_1, a_2, \dots, a_k and a_2, a_3, \dots, a_{k+1} . In other words, if there is a common sequence $\{a_2, \dots, a_k\}$ among the two candidate lattices, then the newly generated sequence of the lattice will be $a_1, a_2, \dots, a_k, a_{k+1}$.

Algorithm 2. Cross-level Itemset

```

1: procedure Cross_level( $E\_SDB_{L_1}, E\_SDB_{L_2}, FCS_{L_1}, FCS_{L_2}, min\_sup$ )
2:   if ( $(E\_SDB_{L_1}.tidList \cap E\_SDB_{L_2}.tidList)$ .count < min_sup) then Exit
3:   for each  $Seq_1$  in  $FCS_{L_1}$  do
4:      $S_1 := Seq_1.Distinct\_Items$ 
5:     for each  $Seq_2.Distinct\_Items$  do
6:        $S_2 := Seq_2.Distinct\_Items$ 
7:        $Conflict\_Set := \{x : x \in S_2 \text{ and } x.Ancessor(L_1) \in S_1\}$ 
8:        $Matching\_Set := \{x : x \in S_2 \text{ and } x \in S_2\}$ 
9:        $UnqSet_1 := S_1 \cap Conflict\_Set.Ancessors(L_1)$ 
10:       $UnqSet_2 := S_2 \cap Conflict\_Set$ 
11:      for each  $SubSet$  in  $Conflict\_Set.Ancessors(L_1)$  do
12:         $ConSet_1 := UnqSet_1 \cup SubSet \cup Matching\_Set$ 
13:         $ConSet_2 := UnqSet_2 \cup SubSet.Descendant$ 
14:         $CombinedSet := ConSet_1 \cup ConSet_2$ 
15:        if  $L_1$ .is SuperSet of ( $L_2$ ) then
16:           $OrderedSet := OrderedSet(CombinedSet, L_1) \triangleright CombinedSet$ 
           elements in such a way that OrderedSet is a subSequence of  $S_1$ 
17:          else  $OrderedSet := OrderedSet(CombinedSet, L_2)$ 
18: procedure Count\_Distict\_Items( $SDB$ )
19:   for each  $item$  in  $SDB$  do
20:     if item not present in DI then  $\triangleright$  DI= Distinct Item
21:        $DI.add(Item)$ 

```

Then, the algorithm 1 constructs cross-level sequential closed pattern using the function *Cross_level*() in line 7 described in Algorithm 2. From two lattices of cardinality k by crossing each tuple of one lattice to all the tuples of another lattice it can generate cross-level closed sequential patterns. Using BIDE in line 5 in Algorithm 1, the resultant tuple T obtained its *TidList* by intersecting the *TidLists* of the candidate tuple pair. The size of the *TidList* of T can be treated as its frequency. Hence, in line 2 in Algorithm 2, the frequency is checked against

the minimum between the two minSups of the candidate lattices. Then if the tuple is found to be frequent, itemsets are calculated for constructing tuples for the resultant lattice at line 3 to line 17.

4 Experimental Results

This section we have evaluated the performance of our proposed approach. The experiments were conducted on a PC with Intel Core i5-3210M CPU at 2.50 GHz and 4 GB RAM. The operating system is Microsoft Windows 10. Our algorithms are implemented using C++ language.

We evaluated our proposed algorithm using some real datasets, including mushroom, chess, retail, pumsp and kosarak. The datasets were obtained from <http://fimi.cs.helsinki.fi/data>.

The general characteristic of the datasets we have used in our work given below:

Table 3. Database characteristics

Database	Total transaction	Avg. size of trans.	Distinct items
Mushroom	8124	23	119
Chess	3195	37	75
Retail	21154	10	15062
Pumsp	49046	74	2113
Kosarak	50462	10	18998

We formed four levels of virtual group in order to create the hierarchical view of the transaction database. The distinct items of the transaction database form the lowest level 4. The number of groups was chosen randomly and we uniformly distributed the items among the groups based on the number of distinct items existing at each level. We varied different parameters, including minimum support and the level of groups, and analyzed the performance of our proposed algorithm CCSP in terms of lattice construction time and number of patterns generated. We performed all of the experiments using the datasets mentioned in Table 3.

4.1 Number of Cross Level Sequential Patterns

To be specific, Mushroom dataset is very dense in which we can get a large number of frequent closed itemsets for all range of values of minimum support. From Fig. 8a the values of the support threshold increase, there is an decrease in the number of overall generated cross-level closed sequential patterns. After crossing among all level we find that huge number of patterns are generated after crossing between the level-2 and level-3 and crossing among level-3 and

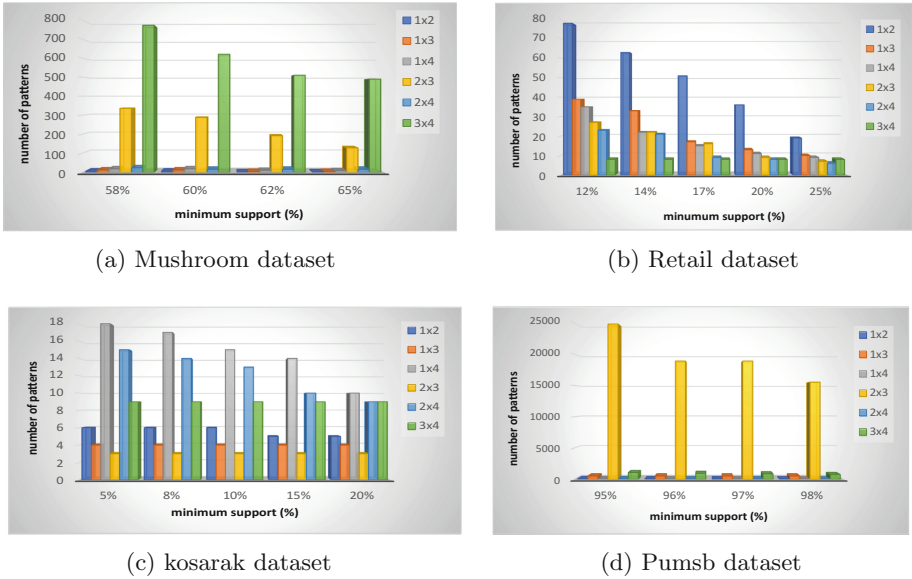


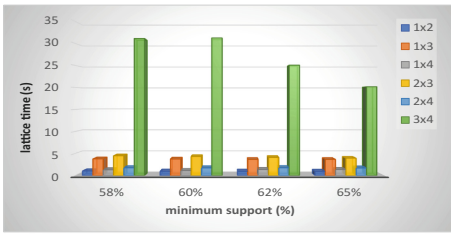
Fig. 8. Number of patterns for different dataset with different values of minimum support for different cross-level

level-4 for mushroom dataset. We obtain a good amount of cross-level sequential closed patterns for the low values of minimum support in Retail Fig. 8b and Kosarak dataset Fig. 8c because of the short length itemset. Pumsb is also of dense characteristics. As the itemset size is long for pumsb dataset, we obtain huge number of cross-level sequential closed patterns for high value of minimum support shown in Fig. 8d.

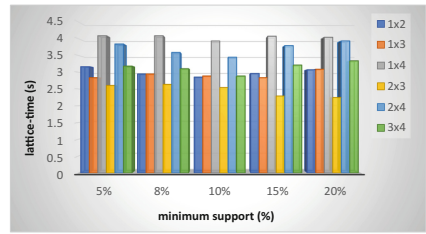
4.2 Run Time for Sequential Closed Lattice Construction

The execution time of our algorithm is composed of lattice construction time and pattern generation time. We plotted time with varying minimum support thresholds. Mushroom dataset is very dense in which we can get a large number of cross-level sequential closed patterns for all range of values of minimum support and so comparatively more time is required for this dataset. Figure 9a shows that more time is required while crossing level-3 level-4 than crossing between other levels. For Chess dataset Fig. 9c shows that the proposed algorithm can find cross-level closed sequential patterns in very short time. The runtime scenario can be visualized for different support threshold for crossing between different levels. Retail database has 21154 transactions and 15062 distinct items but length of transactions is very short. So we obtain good amount of patterns for low values of minimum support. Figure 9d represents that execution time is average because of short length itemset. For the low values of minimum

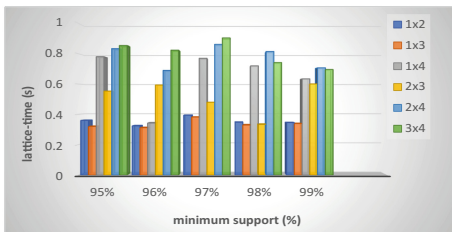
support in Kosarak, the execution time is very low because of the short length itemset which is shown in Fig. 9b.



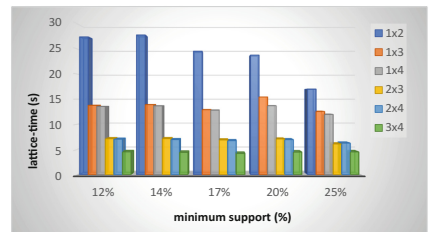
(a) Mushroom dataset



(b) Kosarak dataset

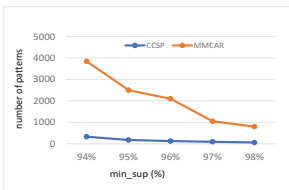


(c) Chess dataset

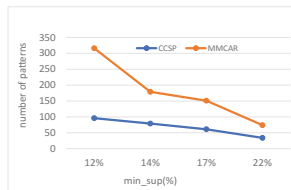


(d) Retail dataset

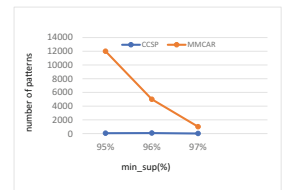
Fig. 9. Lattice construction time for different datasets with different values of minimum support for different cross-level



(a) Chess dataset



(b) Retail dataset



(c) Pumsb dataset

Fig. 10. Graph for CCSP vs. MMCAR based on number of patterns using different support thresholds

4.3 Comparison with Related Work

To compare the performance of our approach CCSP with the existing related approach MMCAR [9], we varied parameter minimum support threshold. The total number of patterns mined by CCSP and MMCAR are plotted for comparison (Fig. 10).

We selected the same range of itemsets for categorizing the items into groups for the two approaches. The comparisons were performed using three dense datasets, Chess, Retail and Pumsb. We considered 3-level hierarchy here. The range of support thresholds values were chosen depending on the dense/sparseness characteristics of the used datasets.

It is observed that with the increase on the threshold values, the number of patterns decreased for all datasets in both CCSP and MMCAR. Our proposed approach mines the cross-level closed patterns that maintain the sequence of items. On the other hand MMCAR mines cross-level closed patterns that do not maintain the sequences. So, we can obtain more informative and interesting knowledge by mining less amount of patterns.

5 Conclusion

In this work, we have designed a new approach to work with an undiscovered area in sequence pattern mining related with hierarchical relationship. We are the first to propose an algorithm for mining the cross-level closed patterns that also maintain the sequence of itemsets from the frequent closed sequential itemset lattice by efficiently traversing the hierarchically built lattices.

Our proposed approach generates closed sequence itemsets at specific levels using BIDE and efficiently generates sequential closed itemsets for the cross-level from the specific level sequential closed itemset lattices and takes very little time in doing so by exploiting the lattice properties of the generated closed sequential itemset patterns. It never generates any multilevel frequent itemsets, rather it directly mines closed sequential itemset at cross-level from the existing closed sequential patterns in a novel way. Since the number of cross-level closed sequential patterns with respect to the cross-level closed patterns is low, our proposed approach CCSP exhibits high performance in comparison with the MMCAR which do not maintain the sequence of itemsets. Using our approach we have mined less number of patterns than MMCAR which provide more interesting, broad and compact yet complete knowledge than MMCAR.

References

1. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the Eleventh International Conference on Data Engineering, pp. 3–14. IEEE (1995)
2. Agrawal, R., Srikant, R., et al.: Fast algorithms for mining association rules. In: Proceeding 20th International Conference very large data bases VLDB, vol. 1215, pp. 487–499 (1994)
3. Ayres, J., Flannick, J., Gehrke, J., Yiu, T.: Sequential pattern mining using a bitmap representation. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 429–435. ACM (2002)
4. Cong, S., Han, J., Padua, D.: Parallel mining of closed sequential patterns. In: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, pp. 562–567. ACM (2005)

5. Gautam, P., Pardasani, K.: A novel approach for discovery multi level fuzzy association rule mining. arXiv preprint [arXiv:1003.4068](https://arxiv.org/abs/1003.4068) (2010)
6. Goethals, B., Zaki, M.J.: Advances in frequent itemset mining implementations: report on fimi'03. *ACM SIGKDD Explor. Newsl.* **6**(1), 109–117 (2004)
7. Han, J., Fu, Y.: Discovery of multiple-level association rules from large databases. In: *VLDB*, vol. 95, pp. 420–431 (1995)
8. Han, J., Fu, Y.: Mining multiple-level association rules in large databases. *IEEE Trans. Knowl. Data Eng.* **11**(5), 798–805 (1999)
9. Hashem, T., Ahmed, C.F., Samiullah, M., Akther, S., Jeong, B.S., Jeon, S.: An efficient approach for mining cross-level closed itemsets and minimal association rules using closed itemset lattices. *Expert Syst. Appl.* **41**(6), 2914–2938 (2014)
10. Leung, C.W.K., Chan, S.C.F., Chung, F.I.: An empirical study of a cross-level association rule mining approach to cold-start recommendations. *Knowl.-Based Syst.* **21**(7), 515–529 (2008)
11. Pei, J., Han, J., Mao, R., et al.: Closet. In: *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*. vol. 4, pp. 21–30 (2000)
12. Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., Hsu, M.C.: Mining sequential patterns by pattern-growth: the prefixspan approach. *IEEE Trans. Knowl. Data Eng.* **16**(11), 1424–1440 (2004)
13. Pokou, Y.J.M., Fournier-Viger, P., Moghrabi, C.: Authorship attribution using small sets of frequent part-of-speech skip-grams. In: *FLAIRS Conference*, pp. 86–91 (2016)
14. Pramono, Y.W.T., et al. In: *2014 International Conference of Advanced Informatics: Concept, Theory and Application (ICAICTA)*, pp. 203–208. IEEE (2014)
15. Shrivastava, V.K., Kumar, P., Pardasani, K.: Fp-tree and cofi based approach for mining of multiple level association rules in large databases. arXiv preprint [arXiv:1003.1821](https://arxiv.org/abs/1003.1821) (2010)
16. Srikant, R., Agrawal, R.: Mining generalized association rules (1995)
17. Srikant, R., Agrawal, R.: Mining sequential patterns: generalizations and performance improvements. In: Apers, P., Bouzeghoub, M., Gardarin, G. (eds.) *EDBT 1996*. LNCS, vol. 1057, pp. 1–17. Springer, Heidelberg (1996). <https://doi.org/10.1007/BFb0014140>
18. Thakur, R.S., Jain, R.C., Pardasani, K.R., India, V.: Mining level-crossing association rules from large databases (2005)
19. Thakur, R., Jain, R., Pardasani, K.: Mining level-crossing association rules from large databases. *J. Comput. Sci.* **2**(1), 76–81 (2006)
20. Wan, Y., Liang, Y., Ding, L.Y.: Mining multilevel association rules with dynamic concept hierarchy. In: *2008 International Conference on Machine Learning and Cybernetics*, vol. 1, pp. 287–292. IEEE (2008)
21. Wang, J., Han, J.: Bide: Efficient mining of frequent closed sequences. In: *Proceedings 20th International Conference on 2004 Data Engineering*, pp. 79–90. IEEE (2004)
22. Wang, J., Han, J., Pei, J.: Closet+: searching for the best strategies for mining frequent closed itemsets. In: *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 236–245. ACM (2003)
23. Yan, X., Han, J., Afshar, R.: Clospan: Mining: closed sequential patterns in large datasets. In: *Proceedings of the 2003 SIAM International Conference on Data Mining*, pp. 166–177. SIAM (2003)
24. Zaki, M.J.: Spade: an efficient algorithm for mining frequent sequences. *Mach. learn.* **42**(1), 31–60 (2001)

25. Zaki, M.J., Hsiao, C.J.: Charm: an efficient algorithm for closed itemset mining. In: Proceedings of the 2002 SIAM International Conference on Data Mining, pp. 457–473. SIAM (2002)
26. Zaki, M.J., Hsiao, C.J.: Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Trans. Knowl. Data Eng.* **17**(4), 462–478 (2005)



An Efficient Approach for Mining Weighted Sequential Patterns in Dynamic Databases

Sabrina Zaman Ishita, Faria Noor, and Chowdhury Farhan Ahmed^(✉)

Department of Computer Science and Engineering,
University of Dhaka, Dhaka, Bangladesh
sz.ishita@gmail.com, faria.noor93@gmail.com, farhan@du.ac.bd

Abstract. In modern time of flowing data where more data accumulate every minute than we can store or make sense of, a fast approach for analyzing incremental or dynamic database has a lot of significance. In a lot of instances, the data are sequential and the ordering of events has interesting meaning itself. Algorithms have been developed to mine sequential patterns efficiently from dynamic databases. However, in real life not all events bear the same urgency or importance, and by treating them as equally important the algorithms will be prone to leaving out rare but high impact events. Our proposed algorithm solves this problem by taking both the weight and frequency of patterns and the dynamic nature of the databases into account. It mines weighted sequential patterns from dynamic databases in efficient manner. Extensive experimental analysis is conducted to evaluate the performance of the proposed algorithm using large datasets. This algorithm is found to outperform previous method for mining weighted sequential patterns when the database is dynamic.

Keywords: Dynamic databases · Weighted sequential pattern
Weighted support · Incremental mining

1 Introduction

Data Mining is the analysis of data, usually in large volumes, for uncovering useful relationships among events or items that make up the data. Frequent pattern mining is an important data mining problem with extensive application; here patterns are mined which occur frequently in a database. Another important domain of data mining is sequential pattern mining where the ordering of items in a sequence is important. Unless weights (value or cost) are assigned to individual items, they are usually treated as equally valuable. However, that is not the case in most real life scenarios. When the weight of items is taken into account in a sequential database, it is known as weighted sequential pattern mining.

As technology and memory devices improve at an exponential rate, their usage grows along too, allowing for the storage of databases to occur at an even higher rate. This calls for the need of incremental mining for dynamic databases

whose data are being continuously added. Most organizations that generate and collect data on a daily basis have unlimited growth. When a database update occurs, mining patterns from scratch is costly with respect to time and memory. It is clearly unfeasible. Several approaches have been adopted to mine sequential patterns in incremental database that avoids mining from scratch. This way, considering the dynamic nature of the database, patterns are mined efficiently. However, the weights of the items are not considered in those approaches.

Consider the scenario of a supermarket that sells a range of products. Each item is assigned a weight value according to the profit it generates per unit. In the classic style of market basket analysis, if we have 5 items {“milk”, “perfume”, “gold”, “detergent”, “pen”} from the data of the store, the sale of each unit of gold is likely to generate a much higher profit than the sales of other items. Gold will therefore bear a high weight. In a practical scenario, the frequency of sale of gold is also going to be much less than other lower weight everyday items such as milk or detergent. If a frequent pattern mining approach only considers the frequency without taking into account the weight, it will miss out on important events which will not be realistic or useful. By taking weight into account we are also able to prune out many low weight items that may appear a few times but are not significant, thus decreasing the overall mining time and memory requirement.

Existing algorithms for mining weighted sequential patterns or mining sequential patterns in incremental database give compelling results in their own domain, but have the following drawbacks: existing sequential pattern mining algorithms in incremental database do not consider weights of patterns, though low-occurrence patterns with high-weight are often interesting, hence they are missed out if uniform weight is assigned. Weighted sequential patterns are mined from scratch every time the database is appended, which is not feasible for any repository that grows incrementally. These motivated us to overcome these problems and provide a solution that gives better result compared to state-of-the-art approaches. In our approach we have developed an algorithm to mine weighted sequential patterns in an incremental database that will benefit a wide range of applications, from Market Transaction and Web Log Analysis to Bioinformatics, Clinical and Network applications.

With this work we have addressed an important sub-domain of frequent pattern mining where several categories such as sequential, weighted and incremental mining collide. Our contributions are: (1) the construction of an algorithm, *WIncSpan*, that is capable of mining weighted sequential patterns in dynamic databases continuously over time. (2) Thorough testing on real life datasets to prove the competence of the algorithm for practical use. (3) Marked improvement in results of the proposed method when compared to existing algorithm.

The paper is organized as follows: Sect. 2 talks about the preliminary concepts and discusses some of the state-of-the-art mining techniques which directly influence this study. In Sect. 3, the proposed algorithm is developed and an example is worked out. Comparison of results of the proposed algorithm with existing algorithm is given in Sect. 4. And finally, the summary is provided as conclusions in Sect. 5.

2 Preliminary Concepts and Related Work

Let us expand our discussion to better understand the concepts that lie at the heart of mining frequent patterns of different types. Let I be the set of all items I_1, I_2, \dots, I_n . A set of transactions is considered as a transaction database where each transaction is a subset of I . Sequence database is a set of sequences where every sequence is a set of events $\langle e_1 e_2 e_3 \dots e_l \rangle$. The order in which events or elements occur is important. Here, event e_1 occurs before event e_2 , which occurs before e_3 and so on. Each event $e_i \subseteq I$.

In a sequence database, support is the count of how frequently a pattern or sequence appears in the database. The support of a pattern P with respect to a sequence database is defined as the number of sequences which contain P . In Table 1, a sequence database is given along with one increment, where in first sequence, there are 2 events: (ab) and (e). For brevity, the brackets are omitted if an event has only one item. Here, (ab) occurs before (e). Given a set of sequences and a user-specified minimum support threshold min_sup , sequential pattern mining is regarded as finding all frequent subsequences whose support count is no less than min_sup . If $\alpha = \langle (ab)b \rangle$ and $\beta = \langle (abc)(be)(de)c \rangle$, where a, b, c, d, and e are items, then α is a subsequence of β .

Many algorithms, such as GSP [13] and SPADE [16], mine frequent sequential patterns. GSP uses Apriori based approach of candidate generate and test. SPADE uses the same approach as GSP but it maps a sequence database into vertical data format unlike GSP. They also obey the antimonotone or downward-closure property that if a sequence does not fulfill the minimum support requirement then none of its super-sequences will be able to fulfill it as well. FreeSpan [7] takes motivation from FP-Growth Tree and mines sequential patterns. SPAM [2] mines sequential patterns using a bitmap representation. PrefixSpan [12] maintains the antimonotone property and uses a prefix-projected pattern growth method to recursively project corresponding postfix subsequences into projected databases.

The usage and improvement of technology and memory devices grow at an exponential rate which means the databases are also growing dynamically. This calls for the need of incremental mining for dynamic databases whose data is being continuously added, such as in shopping transactions, weather sequences and medical records. The naive solution for mining patterns in dynamic database is to mine the updated database from scratch, but this will be inefficient since the newly appended portion of the database is often much smaller than the whole database. To produce frequent sequential patterns from dynamic database in an efficient way, several algorithms [9–11] were proposed. One of the algorithms for mining sequential patterns from dynamic databases is IncSpan [4]. Here, along with frequent sequences, semi-frequent sequences are also saved to be worked on when new increment is added. For buffering semi-frequent sequences along with frequent sequences, a buffer ratio is used. In our approach, we will use this concept for buffering weighted semi-frequent sequences for further use.

Considering the importance or weights of items, several approaches such as WSpan [15], WIP [14], WSM [5] etc. have been proposed for mining weighted

frequent patterns. WSpan mines weighted frequent sequential patterns. Using the weight constraint for mining weighted sequential patterns WSpan [15] uses the prefix projected sequential pattern growth approach. According to WSpan, the weight of a sequence is defined as the average weight of all its items from all the events. For example, using the weight table provided in Table 2, we can calculate the weight of the sequential pattern $P = \langle abc \rangle$ as $W(P) = (0.41 + 0.48 + 0.94)/3 = 0.61$.

There exists many work in the field of weighted sequential pattern mining and in the field of incremental mining of sequential patterns separately. But there has been no complete work in the field of mining weighted sequential patterns in incremental databases. A work [8] has attempted to mine weighted sequential patterns in incremental databases, but no complete details and comparative performance analysis were provided there. We are proposing a new algorithm *WIncSpan* which provides a complete work of how weighted sequential patterns can be generated efficiently from dynamic databases and providing detailed experimental results of its performance.

3 The Proposed Approach

In previous section we discussed the preliminary concepts and existing methods of mining frequent sequential patterns separately in weighted and incremental domains. In this chapter we merge those concepts to propose a new method for weighted sequential pattern mining in incremental databases. A sequence database is given in Table 1. Here, from sequences 10 through 50 represent the initial database D and sequences 60 through 80 represent Δdb which is the new appended part of the whole database D' . The corresponding weights of the items of D' is given in Table 2.

Table 1. Appended database D'

	Sequence ID	Sequences
D	10	$\langle (ab)e \rangle$
	20	$\langle ab \rangle$
	30	$\langle a(dc)e \rangle$
	40	$\langle (ab)d \rangle$
	50	$\langle b(dce) \rangle$
Δdb	60	$\langle (ab)d \rangle$
	70	$\langle a(dc)(ab) \rangle$
	80	$\langle a(ab)e \rangle$

Table 2. Weight table for items

Item	Weight
a	0.41
b	0.48
c	0.94
d	0.31
e	0.10

Definition 1 (Minimum Weighted Support: $minw_sup$). As we know, for a given minimum support percentage, the min_sup value is calculated as:

$min_sup = \text{number of transactions in database} * \text{minimum support percentage}$. We are considering the weight of the items as well, we derive a minimum weighted support threshold $minw_sup$:

$$minw_sup = min_sup * avgW$$

Here, $avgW$ is the average weight value. This is the average of the total weight or profit that has contributed to the database upto that point. In initial database of D' , item a occurs 4 times in total, similarly b occurs 4, c occurs 2, d occurs 3 and e occurs 3 times in total. The $avgW$ is calculated as: $avgW = (4 * 0.41) + (4 * 0.48) + (2 * 0.94) + (3 * 0.31) + (3 * 0.10) / 16 = 0.4169$. In initial database D , the $minw_sup$ for minimum support 60% is therefore calculated as: $minw_sup = 3 * 0.4169 = 1.25$ (as $min_sup = 5 * 60\% = 3$).

Definition 2 (Possible Frequent Sequences). The possible set of frequent sequences is generated to list sequences or patterns in a database that have a chance to grow into patterns that could be frequent later. For a sequence to be possibly frequent, the following condition must be fulfilled:

$$support * maxW \geq minw_sup$$

The notation $maxW$ denotes the weight of the item in the database that has maximum weight. In our example, it would be 0.94 for the item $\langle c \rangle$. This value is multiplied with the support of the pattern instead of taking the actual weight of the pattern. This is to make sure the anti-monotone property is maintained, since in an incremental database a heavy weighted item may appear later on in the same sequence with less weighted items, thereby lifting the overall support of the pattern. By taking the maximum weight, an early consideration is made to allow growth of patterns later on during prefix projection. The set thus contains all the frequent items, as well as some infrequent items that may grow into frequent patterns later, or be pruned out.

Complete Set of Possible Frequent Sequences. First, the possible length-1 items are mined. For item $\langle a \rangle$ in D , $support_a * maxW = 4 * 0.94 = 3.76 \geq minw_sup$. The item $\langle a \rangle$ satisfies the possible frequent sequence condition. Items $\langle b \rangle$, $\langle c \rangle$, $\langle d \rangle$ and $\langle e \rangle$ are found to satisfy the condition as well and therefore are added to the set of possible frequent length-1 sequences.

Possible Frequent length-1 Sequences: $\{\langle a \rangle, \langle b \rangle, \langle c \rangle, \langle d \rangle, \langle e \rangle\}$

Next, the projected database for each frequent length-1 sequence is produced using the frequent length-1 sequence as prefix. The projected databases are mined recursively by identifying the local weighted frequent items at each layer, till there are no more projections. In this way the set of possible frequent sequences is grown, which now includes the sequential patterns grown from the length-1 sequences. At each step of the projection, the items picked will have to satisfy the minimum weighted support condition. For example, for

item $\langle a \rangle$, the projected database contains these sequences: $\langle (_b)e \rangle$, $\langle b \rangle$, $\langle (dc)e \rangle$, $\langle (_b)d \rangle$. And the possible sequential patterns mined from these sequences are: $\langle a \rangle$, $\langle ab \rangle$, $\langle (ab) \rangle$, $\langle ac \rangle$, $\langle ad \rangle$, $\langle ae \rangle$. In the similar way, possible sequential patterns are also mined from the projected databases with prefixes $\langle b \rangle$, $\langle c \rangle$, $\langle d \rangle$ and $\langle e \rangle$.

Definition 3 (Weighted Frequent and Semi-frequent Sequences). For static database, at this moment, only the weighted frequent sequences will be saved and others will be pruned out. Considering the dynamic nature of the database, along with weighted frequent sequences, we will keep the weighted semi-frequent sequences too. From the set of possible frequent sequences, set of Frequent Sequences (*FS*) and Semi-Frequent Sequences (*SFS*) can be constructed as follows:

$$\begin{aligned} \text{Condition for FS: } & \text{support}(P) * \text{weight}(P) \geq \text{minw_sup} \\ \text{Condition for SFS: } & \text{support}(P) * \text{weight}(P) \geq \text{minw_sup} * \mu \end{aligned}$$

Here, P is a possible frequent sequence and μ is a buffer ratio. If the support of P multiplied by its actual weight satisfies the minimum weighted support minw_sup then it goes to the *FS* list. If not, the support of P times its actual weight is compared with a fraction of minw_sup which is derived from multiplying minw_sup by a buffer ratio μ . If satisfied, the sequence is listed in *SFS* as a semi-frequent sequence. Otherwise, it is pruned out.

For example, the single length sequence $\langle a \rangle$ has weighted support $4 * 0.41 = 1.64$. Since 1.64 is greater than the minw_sup value 1.25, $\langle a \rangle$ is added to *FS*. Considering the value of μ as 60%, $\text{minw_sup} * \mu = 1.25 * 60\% = 0.75$. Here, $\langle bd \rangle$ has support count of 2 and weight of $(0.48 + 0.31)/2 = 0.395$. Its weighted support $2 * 0.395 = 0.79$ is greater than 0.75, so it goes to *SFS* list. For initial sequence database D , mined frequent sequences are: $\langle a \rangle$, $\langle b \rangle$, $\langle c \rangle$, $\langle (dc) \rangle$ and semi-frequent sequences are: $\langle (ab) \rangle$, $\langle bd \rangle$, $\langle d \rangle$. Other sequences from possible set of frequent sequences are pruned out as infrequent. Interestingly, we see that $\langle d \rangle$ is a semi-frequent pattern but when we consider it in an event with highly weighted item $\langle c \rangle$, $\langle (dc) \rangle$ becomes a frequent pattern. This is possible in our approach as a result of considering the weight of sequential patterns.

Dynamic Trie Maintenance. An extended trie is constructed from *FS* and *SFS* patterns from D which is illustrated in Fig. 1. The concept of the extended trie is taken from the work [4]. Each node in the trie will be extended from its parent node as either s-extension or i-extension. If the node is added as different event, then it is s-extension, if it is added in the same event as its parent then it is i-extension. For example, while adding the pattern $\langle (ab) \rangle$ to the trie, we first go to the branch labeled with $\langle a \rangle$, increment its support count by the support count of $\langle (ab) \rangle$, then add a new branch to it labeled with $\langle b \rangle$ as i-extension. The solid lines represent the *FS* patterns and the dashed lines represent the *SFS* patterns. Each path from root to non-root node represents a pattern along with its support.

When new increments are added, rather than scanning the whole database to check the new support count of a pattern, the dynamic trie becomes handy. This trie will be used dynamically to update the support count of patterns when new increments will be added to the database. Traversing the trie to get the new support count of a pattern is performed a lot faster than scanning the whole database.

Increment to Database. At this point, if an update to the database is made, which is a common nature of most real-life datasets, it is not convenient to run the procedure from scratch. How the appended part of the database will be handled, how new frequent sequences will be generated using the *FS* and *SFS* lists, how the dynamic trie will be helpful, all are explained below.

The Proposed Algorithm. Here, the basic steps of the proposed *WIncSpan* algorithm is illustrated to mine weighted sequential patterns in an incremental database. Further, an incremental scenario is provided to better comprehend the process.

Snapshot of the Proposed Algorithm. The necessary steps for mining weighted sequential frequent patterns in an incremental database are:

1. In the beginning, the initial database is scanned to form the set of possible frequent patterns.
2. The weighted support of each pattern is compared with the minimum weighted support threshold $minw_sup$ to pick out the actual frequent patterns, which are stored in a frequent sequential set *FS*.
3. If not satisfied, the weighted support of the pattern is checked against a percentage (buffer ratio) of the $minw_sup$ to form the set of semi-frequent set *SFS*. Other patterns are pruned out as infrequent.
4. An extended dynamic trie is constructed using the patterns from *FS* and *SFS* along with their support count.
5. For each increment in the database, the support counts of patterns from the trie are updated.
6. Then the new weighted support of each pattern in *FS* and *SFS* is again compared with the new $minw_sup$ and then compared with the percentage of $minw_sup$ to check whether it goes to new frequent set FS' or to new semi-frequent set SFS' , or it may also become infrequent.
7. FS' and SFS' will serve as *FS* and *SFS* for next increment.
8. At any instance, to get the weighted frequent sequential patterns till that point, the procedure will output the set *FS*.

An Incremental Example Scenario. When an increment to database *D* occurs, it creates a larger database D' as shown in Table 1. Here, three new transactions have been added which is denoted as Δdb .

The $minw_sup$ will get changed due to the changed value of min_sup and $avgW$. Taking 60% as the minimum support threshold as before, new absolute value of $min_sup = 8 * 60\% = 5$ and the new $avgW$ is calculated as 0.422. So, the new $minw_sup$ value is now: $5 * 0.422 = 2.11$. The sequences in Δdb are scanned to check for occurrence of the patterns from FS and SFS , and the support count is updated in the trie. When the support count of patterns in the trie is updated, their weighted support are compared with the new $minw_sup$ and $minw_sup * \mu$ to check if they become frequent or semi-frequent or even infrequent.

After the database update, the newly mined frequent sequences and semi-frequent sequences are listed in Table 3. Patterns not shown in the table are pruned out as infrequent. Although $\langle ab \rangle$ was a semi-frequent pattern in D , it became frequent in D' . On the other hand, the frequent pattern $\langle dc \rangle$ only appears once in Δdb , but it became semi-frequent now. Another pattern, $\langle bd \rangle$, which was semi-frequent in D only increases one time in support in D' . So, $\langle bd \rangle$ falls under the category of infrequent patterns.

Table 3. Weighted frequent and semi-frequent sequences in D'

Frequent sequences	Semi-frequent sequences
$\langle a \rangle, \langle b \rangle, \langle c \rangle, \langle ab \rangle$	$\langle dc \rangle, \langle d \rangle$

After taking the patterns from Δdb into account, the updated FS' and SFS' trie that emerges is illustrated in Fig. 2.

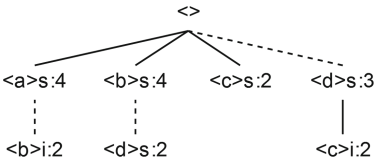


Fig. 1. The sequential pattern trie of FS and SFS in D

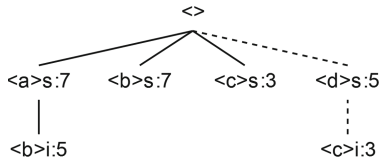


Fig. 2. The updated sequential pattern trie of FS' and SFS' in D'

3.1 The Pseudo-code

To get the weighted sequential patterns from the given databases which are dynamic in nature, we will use the proposed $WIncSpan$ algorithm. A sequence database D , the minimum weighted support threshold $minw_sup$ and the buffer ratio μ are given as input to the algorithm. Algorithm $WIncSpan$ will generate the set of weighted frequent sequential patterns at any instance. The pseudo-code is given in Algorithm 1.

Algorithm 1. WIncSpan: Weighted Sequential Pattern Mining in Dynamic Database

Input: A sequence database D , the minimum weighted support threshold $minw_sup$ and the buffer ratio μ

Output: The set of weighted frequent sequential patterns FS.

Method:

Begin

1. Let WSP be the set of Possible Weighted Frequent Sequential Patterns, FS be the set of Frequent Patterns and SFS be the set of Semi-Frequent Patterns.
Now,
 $WSP \leftarrow \{\}, FS \leftarrow \{\}, SFS \leftarrow \{\}$
 2. $WSP =$ Call the modified WSpan($WSP, D, minw_sup$)
 3. **for** each pattern P in WSP **do**
 4. **if** $sup(P) * weight(P) \geq minw_sup$ **then**
 5. insert (FS, P)
 6. **else if** $sup(P) * weight(P) \geq minw_sup * \mu$ **then**
 7. insert (SFS, P)
 8. **end if**
 9. **end for**
 10. **for** each new increment Δdb in D **do**
 11. $FS, SFS =$ Call WIncSpan($FS, SFS, \Delta db, minw_sup, \mu$)
 12. output FS
 13. **end for**
- End

Procedure: WIncSpan($FS, SFS, \Delta db, minw_sup, \mu$)

Parameters: FS : Frequent Sequences upto now; SFS : Semi-Frequent Sequences upto now; Δdb : incremented portion of D ; $minw_sup$: minimum weighted support threshold; μ : buffer ratio.

1. Let FS' and SFS' be the set of new frequent and semi-frequent patterns respectively.
 2. Initialize $FS' \leftarrow \{\}, SFS' \leftarrow \{\}$
 3. **for** each pattern P in FS or SFS **do**
 4. check $\Delta sup(P)$
 5. $sup(P) = sup_D(P) + \Delta sup(P)$
 6. **if** $sup(P) * weight(P) \geq minw_sup$ **then**
 7. insert(FS', P)
 8. **else if** $sup(P) * weight(P) \geq minw_sup * \mu$ **then**
 9. insert (SFS', P)
 10. **end if**
 11. **end for**
 12. return FS', SFS'
-

In the algorithm, the main method creates the possible set of weighted sequential patterns by calling the modified WSpan as per above discussion. And from that set, the set of frequent sequential patterns FS and the set of

semi-frequent sequential patterns SFS are created according to pre-defined conditions. Each increment is handled in this method. For each of the increment, Procedure: $WIncSpan$ is called with necessary parameters. It creates the set of new frequent and semi-frequent sequential patterns FS' and SFS' respectively. Here, the support count of each pattern of FS and SFS is updated and then the pattern goes to either FS' or SFS' based on two conditions provided. It returns the complete FS' and SFS' which are to be used for further increments.

At any instance, we can check the FS list to get the weighted frequent sequential patterns till that point.

4 Performance Evaluation

In this section, we present the overall performance of our proposed algorithm $WIncSpan$ over several datasets. The performance of our algorithm $WIncSpan$ is compared with $WSpan$ [15]. Various real-life datasets such as SIGN, BIBLE, Kosarak etc. were used in our experiment. These datasets were in spmf [6] format. Some datasets were collected directly from their site, some were collected from the site Frequent Itemset Mining Dataset repository [3] and then converted to spmf format. Both of the implementations of $WIncSpan$ and $WSpan$ were performed in Windows environment (Windows 10), on a core-i5 intel processor which operates at 3.2 GHz with 8 GB of memory.

Using real values of weights of items might be cumbersome in calculations. We used normalized values instead. To produce normalized weights, normal distribution is used with a suitable mean deviation and standard deviation. Thus the actual weights are adjusted to fit the common scale. In real life, items with high weights or costs appear less in number. So do the items with very low weights. On the other hand items with medium range of weights appear the most in number. To keep this realistic nature of items, we are using normal distribution for weight generation.

Here, we are providing the experimental results of the $WIncSpan$ algorithm under various performance metrics. Except for the scalability test, for other performance metrics, we have taken an initial dataset to apply $WIncSpan$ and $WSpan$, then we have added increments in the dataset in two consecutive phases. To calculate the overall performance of both of the algorithms, we measured their performances in three phases.

Performance Analysis w.r.t Runtime. We measured the runtime of $WIncSpan$ and $WSpan$ in three phases. The graphical representations of runtime with varying min_sup threshold for BMS2, BIBLE and Kosarak datasets are shown in Figs. 3, 4 and 5 respectively. Like sparse dataset as Kosarak, the runtime performance was also observed on dense dataset as SIGN. Figure 6 shows the graphical representation.

In the figures, we can see that the time required to run $WIncSpan$ is less than the time required to run $WSpan$. And their differences in time becomes larger when the minimum support threshold is lowered. To understand how the

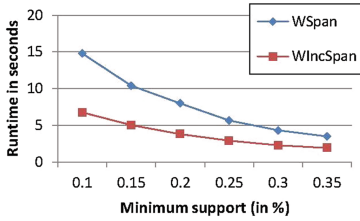


Fig. 3. Runtime for varying min_sup in BMS2 dataset.

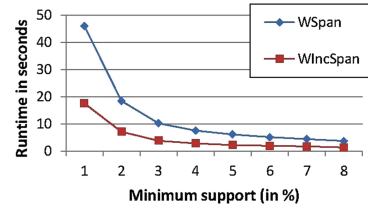


Fig. 4. Runtime for varying min_sup in BIBLE dataset.

runtime calculation is done more clearly, Table 4 shows the runtime in each phase for both *WIncSpan* and *WSpan* in Kosarak dataset. The total runtime is calculated which is used in the graph. It is clear that *WIncSpan* outperforms *WSpan* with respect to runtime. With the dynamic increment to the dataset, it is desirable that we generate patterns as fast as we can. *WIncSpan* fulfills this desire, and it runs a magnitude faster than *WSpan*.

Table 4. Runtime performance of *WSpan* and *WIncSpan* with varying min_sup in Kosarak dataset

min_sup (in %)	Runtime in initial database		Runtime after 1 st increment		Runtime after 2 nd increment		Runtime total (in seconds)	
	WSpan	WIncSpan	WSpan	WIncSpan	WSpan	WIncSpan	WSpan	WIncSpan
0.22%	81.3	81.3	66.8	1.66	65.5	1.53	213.6	84.49
0.26%	22.8	22.8	38.4	1.56	51.1	0.82	108.3	25.18
0.3%	14.5	14.5	20.3	0.75	26.8	0.63	61.6	15.88
0.34%	4.63	4.63	8.12	0.56	15.1	0.48	27.85	5.67
0.38%	2.11	2.11	3.11	0.48	5.11	0.54	10.33	3.13

Performance Analysis w.r.t Number of Patterns. The comparative performance analysis of *WIncSpan* and *WSpan* with respect to number of patterns for Kosarak and SIGN datasets are given in Figs. 7 and 8 respectively. In these graphs, we can see that the number of patterns generated by *WSpan* is more than the number of patterns generated by *WIncSpan*. As the minimum threshold is lowered, this difference gets bigger. However, the advantage of *WIncSpan* over *WSpan* is that it can generate these patterns way faster than *WSpan* as we saw in the previous section.

Performance Analysis with Varying Buffer Ratio. The lower the buffer ratio is, the higher the buffer size, which can accommodate more semi-frequent patterns. We have measured the number of patterns by varying the buffer ratio.

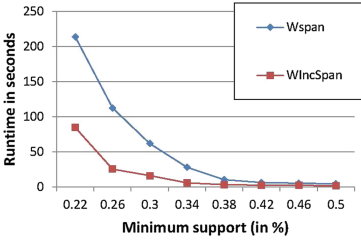


Fig. 5. Runtime for varying min_sup in Kosarak dataset.

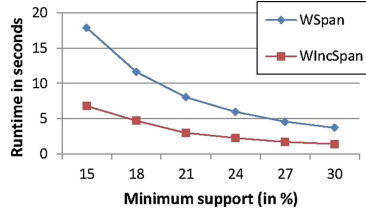


Fig. 6. Runtime for varying min_sup in SIGN dataset.

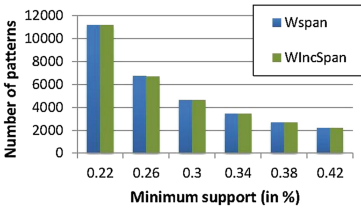


Fig. 7. Number of patterns for varying min_sup in Kosarak dataset.

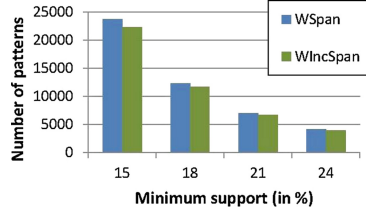


Fig. 8. Number of patterns for varying min_sup in SIGN dataset.

The graphical representation of the results in BIBLE dataset is shown in Fig. 9. Here, we can see that by increasing the buffer ratio the number of patterns tends to decrease. Because smaller number of semi-frequent patterns are generated and they can contribute less to frequent patterns in the next phase. We can also see that the number of patterns generated by WSpan is constant for several buffer ratio because WSpan does not buffer semi-frequent patterns, it generates patterns from scratch in every phase.

Performance Analysis w.r.t Memory. Figure 10 shows memory consumption by both *WIncSpan* and *WSpan* with varying min_sup in Kosarak dataset. For every dataset, it showed that memory consumed by *WIncSpan* is lower than memory consumed by *WSpan*. This is because *WIncSpan* scans the new appended part of the database and works on the dynamic trie. Whereas *WSpan* creates projected database for each pattern and generates new patterns from it. This requires a lot more memory compared with *WIncSpan*.

Performance Analysis with Varying Standard Deviation. To generate weights for items, we have used normal distribution with a fixed mean deviation of 0.5 and varying standard deviation (0.15 in most of the cases). For varying standard deviation, the number of items versus weight ranges curves are shown in Fig. 11. The range (mean deviation \pm standard deviation) holds the most amount of items which is the characteristic of real-life items. In real life, items with medium values occur frequently whereas items with higher or too lower values occur infrequently.

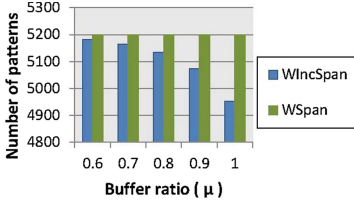


Fig. 9. Number of patterns for varying buffer ratio (μ) in BIBLE dataset.

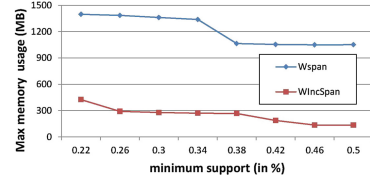


Fig. 10. Memory usage for varying min_sup in Kosarak dataset.

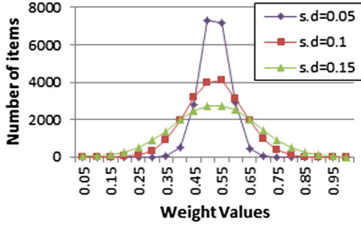


Fig. 11. Weight values by normal distribution with different standard deviations in Kosarak dataset.

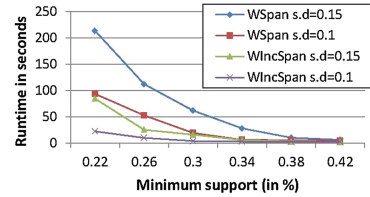


Fig. 12. Runtime evaluation with different standard deviations in Kosarak dataset.

Figures 12 and 13 show performance of *WIncSpan* and *WSpan* with respect to runtime and number of patterns respectively with different standard deviations. *WIncSpan* outperforms *WSpan* in case of runtime. The number of patterns in each case does not differ a lot from each other. So, it is clear that *WIncSpan* can work better than *WSpan* with varying weight ranges too.

Scalability Test. To test whether *WIncSpan* is scalable or not, we have run it on different datasets with several increments. Figure 14 shows the scalability performance analysis of *WIncSpan* and *WSpan* in Kosarak dataset when the minimum support threshold is 0.3%. After running on an initial set of the database, five consecutive increments were added and the runtime performance was measured in each step.

Here, we can see that both *WSpan* and *WIncSpan* take same amount of time in initial set of database. As the database grows dynamically, *WSpan* takes more time than *WIncSpan*. *WIncSpan* tends to consume less time from second increment as it uses the dynamic trie and new appended part of the database only. From second increment to the last increment, consumed time by *WIncSpan* does not vary that much from each other. So, we can see that *WIncSpan* is scalable along with its runtime and memory efficiency.

The above discussion implies that *WIncSpan* can be applied in real life applications where the database tends to grow dynamically and the values (weights) of the items are important. *WIncSpan* outperforms *WSpan* in all the cases. In case of number of weight patterns, *WIncSpan* may provide less amount of patterns than

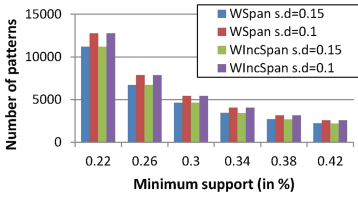


Fig. 13. Number of patterns for varying standard deviations in Kosarak dataset.

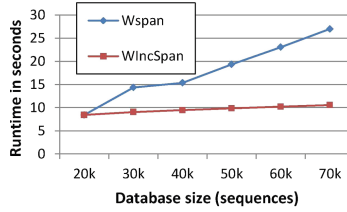


Fig. 14. Scalability test in Kosarak dataset ($\text{min_sup} = 0.3\%$)

WSpan, but this behaviour can be acceptable considering the remarkable less amount of time it consumes. In real life, items with lower and higher values are not equally important. So, *WIncSpan* can be applied in place of *IncSpan* also where the value of the item is important.

5 Conclusions

A new algorithm *WIncSpan*, for mining weighted sequential patterns in large incremental databases, is proposed in this study. It overcomes the limitations of previously existing algorithms. By buffering semi-frequent sequences and maintaining dynamic trie, our approach works efficiently in mining when the database grows dynamically. The actual benefits of the proposed approach is found in its experimental results, where the *WIncSpan* algorithm has been found to outperform WSpan. It is found to be more time and memory efficient.

This work will be highly applicable in mining weighted sequential patterns in databases where constantly new updates are available, and where the individual items can be attributed with weight values to distinguish between them. Areas of application therefore include mining Market Transactions, Weather Forecast, improving Health-Care and Health Insurance and many others. It can also be used in Fraud Detection by assigning high weight values to previously found fraud patterns.

The work presented here can be extended to include more research problems to be solved for efficient solutions. Incremental mining can be done on closed sequential patterns with weights. It can also be extended for mining sliding window based weighted sequential patterns over datastreams [1].

References

1. Ahmed, C.F., Tanbeer, S.K., Jeong, B.S., Lee, Y.K.: An efficient algorithm for sliding window-based weighted frequent pattern mining over data streams. *IEICE Trans. Inf. Syst.* **92**(7), 1369–1381 (2009)
2. Ayres, J., Flannick, J., Gehrke, J., Yiu, T.: Sequential pattern mining using a bitmap representation. In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2002*, pp. 429–435. ACM, New York (2002)

3. Bart, G., Mohammed, J.Z.: Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, FIMI 2004, Brighton, UK, 1 November 2004. FIMI (2005). <http://fimi.ua.ac.be/>
4. Cheng, H., Yan, X., Han, J.: IncSpan: incremental mining of sequential patterns in large database. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 527–532. ACM (2004)
5. Cui, W., An, H.: Discovering interesting sequential pattern in large sequence database. In: Asia-Pacific Conference on Computational Intelligence and Industrial Applications, PACIIA 2009, vol. 2, pp. 270–273. IEEE (2009)
6. Fournier-Viger, P., et al.: The SPMF open-source data mining library version 2. In: Berendt, B., et al. (eds.) ECML PKDD 2016. LNCS, vol. 9853, pp. 36–40. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46131-1_8
7. Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., Hsu, M.C.: FreeSpan: frequent pattern-projected sequential pattern mining. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 355–359. ACM (2000)
8. Kollu, A., Kotakonda, V.K.: Incremental mining of sequential patterns using weights. *IOSR J. Comput. Eng.* **5**, 70–73 (2013)
9. Lin, J.C.W., Hong, T.P., Gan, W., Chen, H.Y., Li, S.T.: Incrementally updating the discovered sequential patterns based on pre-large concept. *Intell. Data Anal.* **19**(5), 1071–1089 (2015)
10. Massegli, F., Poncet, P., Teisseire, M.: Incremental mining of sequential patterns in large databases. *Data Knowl. Eng.* **46**(1), 97–121 (2003)
11. Nguyen, S.N., Sun, X., Orłowska, M.E.: Improvements of IncSpan: incremental mining of sequential patterns in large database. In: Ho, T.B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS, vol. 3518, pp. 442–451. Springer, Heidelberg (2005). https://doi.org/10.1007/11430919_52
12. Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., Hsu, M.C.: Mining sequential patterns by pattern-growth: the PrefixSpan approach. *IEEE Trans. Knowl. Data Eng.* **16**(11), 1424–1440 (2004)
13. Srikant, R., Agrawal, R.: Mining sequential patterns: generalizations and performance improvements. In: Apers, P., Bouzeghoub, M., Gardarin, G. (eds.) EDBT 1996. LNCS, vol. 1057, pp. 3–17. Springer, Heidelberg (1996). <https://doi.org/10.1007/BFb0014140>
14. Yun, U.: Efficient mining of weighted interesting patterns with a strong weight and/or support affinity. *Inf. Sci.* **177**(17), 3477–3499 (2007)
15. Yun, U.: A new framework for detecting weighted sequential patterns in large sequence databases. *Knowl.-Based Syst.* **21**(2), 110–122 (2008)
16. Zaki, M.J.: SPADE: an efficient algorithm for mining frequent sequences. *Mach. Learn.* **42**, 31–60 (2001)



A Decision Rule Based Approach to Generational Feature Selection

Wiesław Paja^(✉)

Faculty of Mathematics and Natural Sciences, University of Rzeszów,
1 Pigonia Street, 35-310 Rzeszów, Poland
wpaja@ur.edu.pl

Abstract. The increase of dimensionality of data is a target for many existing feature selection methods with respect to efficiency and effectiveness. In this paper, the all relevant feature selection method based on information gathered using generational feature selection is described. The successive generations of feature subset were defined using *Rule Quality Importance* algorithm and next the subset of most important features was eliminated from the primary dataset. This process was executed until the most relevant feature has got importance value on the level equal to importance of the random, shadow feature. The proposed approach was also tested on well-known artificial Madelon dataset and the results confirm its efficiency. Thus, the conclusion is that the identified features are relevant but not all weakly relevant features were discovered.

Keywords: Decision rules · Generational feature selection
Feature ranking · All relevant feature selection
Dimensionality reduction

1 Introduction

In the era of high-dimensional datasets some methods that provide possibilities to effective analysis of such kind of data are crucial. These kind of methods are applied to improve performance in terms of speed, predictive power and simplicity of the model [1]. Moreover, they are used to visualize the data for model selection, to reduce dimensionality and remove noise of themselves. For this reason a feature selection methods are required. The feature selection is a process that able to choose an optimal subset of features according to a certain criterion. In this way, irrelevant data (features) could be removed from the original set. It able to increase the predictive accuracy of developed learning models, reduce the cost of the data. It also can improve the learning efficiency, such as reducing storage requirements and computational cost. Finally, it could be observed the reduction of the complexity of the resulting model description, improving the understanding of the data and the model. Feature selection can be considered as a search problem, where each state of the search space corresponds to

a concrete subset of features selected [2]. Thus, two main directions of search can be identified: *Sequential Forward Generation (SFG)* and *Sequential Backward Generation (SBG)*. According to *SFG* approach searching process starts with an empty set of features F . As the search starts, features are added into F according to some criterion that distinguish the relevant feature from the others. Features set F grows until it reaches some stopping criteria. It can be a threshold for the number of relevant features m or simply the generation of all possible subsets in brute force mode. In turn, the *SFG* approach searching process starts with a full set of features and, iteratively, they are removed one at a time. Here, the criterion must point out the least relevant feature. Finally, the subset contains only a unique feature, which is considered to be the most informative of the whole set. As in the previous case, different stopping criteria can be used. The two other direction of search can be also recognized: the first one *Bidirectional Generation (BG)* begins the search in both directions, performing *SFG* and *SBG* concurrently. They stop in two cases: when one search finds the best subset comprised of m features before it reaches the exact middle, or both searches achieve the middle of the search space. It takes advantage of both *SFG* and *SBG*. The second method *Random Generation (RG)* starts the search in a random direction. The choice of adding or removing a features is a random decision. It tries to avoid the stagnation into a local optima by not following a fixed way for subset generation. Unlike *SFG* or *SBG*, the size of the subset of features cannot be stipulated.

During the selection process two main goals are identified [3]:

- *Minimal Optimal Feature Selection (MOSF)*, where the goal is to discover the minimal subset of features with the best classification quality;
- *All Relevant Feature Selection (ARFS)*, where the main goal is to discover all informative features, even that with minimal relevance [4,5];

Here, presented approach is focused on the second type of FS. Motivation for this methodology was *Recursive Feature Elimination (RFE)* algorithm in which by application of external estimator specific weights values are assigned to each features [6,7]. This procedure is repeated recursively, and in each step, features whose weights are the smallest are removed from the currently investigated set of features. It works until the expected set of features to select is eventually obtained. In *RFE* approach a number of feature to select should be initially defined. In turn, in presented approach, the number of feature is unknown and to distinguish between relevant and irrelevant features the *contrast variable* concept [5] has been applied. Contrast does not carry information on the decision attribute by design but it is added to the system in order to discern relevant and irrelevant (shadow) attributes. Contrast values are obtained from the original features by random permutation of values through the objects of the analyzed dataset. The use of contrast variables was for the first time proposed by Stopiglia [8] and then by Tuv [9]. In this way, the goal of proposed methodology is to simplify and improve feature selection process by relevant feature selection during recursive generation of decision rules. The hypothesis is that by removing

subsets of relevant features in each step gradually all-relevant feature subset could be indicated.

2 Methods and Algorithms

In this section, theoretical description of applied methods and algorithms is shortly presented. Mainly, the *Generational Feature Selection* approach and connected methods for contrast feature generation and decision rule quality based importance estimation are described.

2.1 The Decision Rule Quality Based Importance Algorithm

During research the *DRQualityImp* algorithm [10,11] is used to define ranking values for each investigated feature. This algorithm (see *Algorithm 1*) is based on the presence of different feature in decision rule set generated from dataset. Thus, the ranking measure *RQI* (*Rule Quality based Importance*) for attribute a could be defined.

$$RQI(a) = \sum_{j=1}^k q_j \cdot \omega(a) \quad (1)$$

where: k is a number of rules generated; $\omega(a)$ describes the occurrence of the attribute a in j -th rule and q_j denotes the quality of this rule. The pseudocode for this algorithm is presented below. Here, as an input, a special case of a decision system is considered, when only one decision attribute d which determining classes of objects in the decision system S is distinguished, i.e., $S = (U, A \cup \{d\})$. U is the nonempty finite set of objects known as the universe of discourse and

Algorithm 1. Decision Rule Quality based Importance estimation

Input : $S = (U, A \cup \{d\})$ - a decision system; R - a decision rule set; Q - quality of rules $r \in R$, $cond(r)$ - a set of conditionals of the rule r .

Output: RQI - a set of importances of features.

Function *ruleQualityImportance*(A, R, Q)

```

    RQI ← ϕ
    for each a ∈ A do
        RQI(a) = 0
        for each r ∈ R do
            if a ⊂ cond(r) then
                RQI(a) = RQI(a) + Q(r)
            end
        end
        RQI ← RQI ∪ RQI(a)
    end
    return RQI in decreasing order

```

end

Algorithm 2. Contrast features generation

Input : $S = (U, A \cap \{d\})$ - a decision system.
Output: A_{CONT} - a set of contrast features.
Function *contrastFeatures*(A)

```

     $A_{CONT} \leftarrow \phi$ 
    for each  $a \in A$  do
         $a_{CONT} \leftarrow \text{permute}(V_a)$ 
         $A_{CONT} \leftarrow A_{CONT} \cup a_{CONT}$ 
    end
    return  $A_{CONT}$ 
end
```

A is the nonempty finite set of attributes (features). Each attribute $a \in A$ is a function $a : U \rightarrow V_a$, where V_a is the set of values of a .

2.2 The Contrast Features Generation Algorithm

Presented approach applies also contrast features algorithm (see *Algorithm 2*) that is used to establish threshold between relevant and irrelevant features inside the investigated set. Here, the decision system $S = (U, A \cup \{d\})$ is also considered as the input. Contrast (shadow) features set A_{CONT} are generated from original set A by the random permutation of each attribute a through their values space V_a . So, as the output the set of contrast attributes is obtained. It is assumed, that these features are not correlated with decision one, and in this way some kind of artificial noise could be added to original data.

2.3 The Generational Feature Selection Algorithm

The experimental procedure, initially called *Generational Feature Selection*, is presented in the form of pseudocode bellow as the *Algorithm 3*. As an input, the decision system $S = (U, A \cup \{d\})$ is considered. Additionally, machine learning algorithm *MLA* utilized during feature importance estimation has to be introduced. The output is in the form of selected important features subset *FS*. Generally, algorithm iteratively generates learning model thus it could be called *generation*. After the first generation (iteration) selected important features are removed from dataset. The next generation is done based on the remaining data, and so on. During each iteration the contrast features A_{CONT} are generated (*contrastFeatures*) based on original current set A_{CURR} and added to it creating extended set A_{EXT} . Using this extended set the learning model m is developed (*generateModel*) and machine learning algorithm *MLA* is applied. Here, decision rule algorithm is utilized. Then, the set of importances M of each feature in developed learning model m is calculated (*modelMeasure*), and selected ranking algorithm is applied (*rankingAlgorithm*) to obtain ranking L . During research it was *ruleQualityImportance* method (see *Algorithm 1*). Next, contrast feature

Algorithm 3. Generational Feature Selection

Input : $S = (U, A \cap \{d\})$ - a decision system; MLA - an applied machine learning algorithm.

Output: FS - a selected feature subset.

Function $GFS(S, MLA)$

```

   $ACURR \leftarrow A$ 
   $FS \leftarrow \phi$ 
   $x=0$ 
  while  $x=0$  do
     $ACONT \leftarrow contrastFeatures(ACURR)$ 
     $AEXT \leftarrow ACURR \cup ACONT$ 
     $m \leftarrow generateModel(S, MLA)$ 
     $M \leftarrow modelMeasure(m)$ 
     $L \leftarrow rankingAlgorithm(AEXT, M)$ 
     $maxCFRank \leftarrow max(L(a : a \in ACONT))$ 
     $FS \leftarrow \phi$ 
    for each  $l \in L(a : a \in ACURR)$  do
      if  $l > maxCFRank$  then
         $FS \leftarrow FS \cup a(l)$ 
      end
       $ACURR \leftarrow ACURR \setminus FS$ 
      if  $FS = \phi$  then
         $x++$ 
      end
       $FS \leftarrow A \setminus ACURR$ 
    end
  end
  return  $FS$ 
end

```

with maximal value of ranking ($maxCFRank$) is identified. After that, set of relevant features FS which have ranking value l higher than $maxCFRank$ is identified. Discovered FS set is then removed from the currently investigated set $ACURR$. If FS is empty then iterations stop. Finally, FS is defined by removing irrelevant features from the original set A .

3 Results and Conclusions

For illustration of the test of proposed algorithm the well-known in the domain of feature selection Madelon dataset is considered. It is an artificial data set, which was one of the Neural Information Processing Systems challenge problems in 2003 (called NIPS2003) [12]. It contains 2600 objects (2000 of training objects + 600 of validation objects) corresponding to points located in 32 vertices of a 5-dimensional hypercube. Each vertex is randomly assigned to the one of two classes: -1 or $+1$, and the decision of each object is a class of its vertex.

Objects are characterized by 500 features which were constructed in the following way: 5 of them are randomly jittered coordinates of points, the other 15 attributes are random linear combinations of the first 5. The rest of the data is a uniform random noise. The goal is to select 20 important attributes from the system without false attributes selection. Additionally, the same 10 fold cross validation process was applied during experiments to efficient comparison of different approaches. Here, only detailed results for 6th sample fold of the Madelon dataset using *RQI* method based on the *CN2* rule quality or own rule quality measure with the *WRACC* evaluation function (see Tables 1 and 2) and also for 2nd fold with the *Laplace* evaluation function (see Table 3) are presented. As it was shown for example in Table 1, four iterations (generations) of the algorithm were done. During the first iteration, the classification rules (the first generation of rules) have been built based on all input data. Using the *RQI* method based on *CN2* rule quality measure, the subset of fourteen features is indicated as relevant one (grey cells marked), according to decreased values of the *RQI* calculated from the developed decision rules set. Then, the subset of selected features is removed from the dataset. Next, in the 2nd iteration of algorithm the next one, probably relevant, subset is selected using the *RQI* values calculated from the rules developed on the reduced dataset. The second feature, *f203_1*, which is the contrast feature defines the threshold for selection of the important subset. This subset (just one feature) is also removed from dataset. Next, in the

Table 1. The sample results of importance of features gathered in 6th fold using *RQI* based on *CN2* rule quality measure and the *WRACC* evaluation function. Bold names denote truly relevant features, others denote irrelevant features. Name with *_1* index denote contrast feature. The grey colored cells denote the feature set found important in given iteration (generation) and which is removed from the data in the next iteration.

1st iteration		2nd iteration		3rd iteration		4th iteration	
feature name	RQI CN2	feature name	RQI CN2	feature name	RQI CN2	feature name	RQI CN2
f339	0.225	f379	0.035	f5	0.071	f213_1	0.037
f337	0.176	f203_1	0.021	f190	0.059	f31_1	0.035
f242	0.151	f411	0.021	f191	0.056	f186	0.021
f65	0.108	f315	0.021	f135_1	0.053	f73_1	0.020
f473	0.090	f286	0.021	f357	0.033	f53_1	0.016
f443	0.087	f217	0.020	f366	0.032	f249	0.016
f494	0.085	f458	0.020	f485_1	0.032	f357	0.015
f454	0.077	f154	0.019	f327_1	0.032	f205_1	0.015
f476	0.057	f435_1	0.019	f83_1	0.030	f497	0.015
f49	0.051	f497_1	0.019	f40	0.030	f328_1	0.015
f129	0.050	f282	0.017	f166	0.030		
f456	0.048	f5	0.017	f183_1	0.029		
f106	0.035	f357	0.016	f125_1	0.029		
f319	0.025				

Table 2. The sample results of importance of features gathered in 6th fold using *RQI* based on *own rule quality measure* and the *WRACC evaluation function*. Bold names denote truly relevant features, others denote irrelevant features. Name with *_1* index denote contrast feature. The grey colored cells denote the feature set found important in given iteration (generation) and which is removed from the data in the next iteration.

1st iteration		2nd iteration		3rd iteration		4th iteration	
feature name	RQI CN2	feature name	RQI CN2	feature name	RQI CN2	feature name	RQI CN2
f339	2.970	f379	1.624	f190	2.742	f31_1	1.237
f337	2.616	f154	0.84	f5	2.292	f213_1	1.149
f494	2.002	f282	0.775	f432	1.909	f357	0.742
f242	1.996	f203_1	0.73	f435_1	1.909	f328_1	0.742
f65	1.924	f411	0.73	f448	1.786	f53_1	0.713
f473	1.375	f217	0.675	f183_1	1.783	f249	0.713
f443	1.294	f458	0.675	f11_1	1.779	f205_1	0.709
f454	1.119	f315	0.674	f478_1	1.773	f497	0.709
f49	1.095	f286	0.674	f73_1	0.667
f319	0.857	f435_1	0.649			f186	0.633
f476	0.829	f497_1	0.649				
f106	0.742	f5	0.577				
f129	0.711	f357	0.541				
f456	0.632						

3rd iteration of algorithm the next subset of three probably relevant features is found using the *RQI* values calculated from the rules constructed on the subsequently reduced dataset. The fourth feature, *f135_1*, defines the threshold for selection of the next important subset. This subset is therefore removed from dataset. Finally, in the 4th iteration the subset of important features is empty, because the highest value of the *RQI* measure is reached by contrast feature *f213_1*. In this way, the algorithm stops, and the subset of 18 features is defined as the relevant one. The truly relevant features in Madelon dataset are written in bold. It could be observed that three attributes: *f5*, *f190* and *f191*, were also included in the discovered subset. However their relevance is very random and unique what is simply presented in Table 4, where these attributes are only 3, 2 and 1 times selected respectively. They reach >0.5 of the feature removing probability threshold P_{rm} during the 10-fold cross-validation. Similar results are presented in Tables 2 and 3. Proposed threshold P_{rm} for a given feature *a* is defined below.

$$P_{rm}(a) = \frac{ncross - nsel(a)}{ncross} \tag{2}$$

Here, *ncross* means the number of the validation folds, in turn *n sel* means the number of selections of the feature *a* [13].

Table 3. The sample results of importance of features gathered in 2nd fold using *RQI* based on *CN2* and *own rule quality measure* and the *Laplace evaluation function*. Bold names denote truly relevant features, others denote irrelevant features. Name with *_1* index denote contrast feature. The grey colored cells denote the feature set found important in given iteration (generation) and which is removed from the data in the next iteration.

1st iteration			2nd iteration			3rd iteration			4th iteration			5th iteration		
feat. name	RQI CN2	RQI own	feat. name	RQI CN2	RQI own	feat. name	RQI CN2	RQI own	feat. name	RQI CN2	RQI own	feat. name	RQI CN2	RQI own
f339	37.19	40	f106	11.20	12	f5	5.56	6	f39	5.44	6	f104_1	4.55	5
f476	12.21	13	f443	7.47	8	f393_1	5.09	6	f229_1	4.48	5	f244	3.69	4
f242	12.11	13	f494	6.67	7	f342_1	4.57	5	f84_1	3.77	4	f264	3.69	4
f49	9.40	10	f473	4.72	5	f479_1	4.46	5	f375_1	3.75	4	f411	3.68	4
f454	9.39	10	f229_1	4.55	5	f395_1	4.43	5	f270_1	3.74	4	f269_1	3.68	4
f129	7.26	8	f228	4.51	5	f267_1	3.77	4	f141	3.64	4	f74_1	3.68	4
f337	6.61	7	f229	4.41	5	f291_1	3.76	4	f310	3.64	4	f291_1	3.66	4
f379	6.48	7	f405_1	3.98	5	f54_1	3.71	4	f387	3.64	4	f46_1	3.62	4
f65	4.44	5	f319	3.82	4	f206	3.69	4	f492_1	3.50	4	f357	3.61	4
f174	4.42	5	f282	3.80	4
f29	3.84	4	f5	3.73	4									
f466_1	3.69	4									
...												

The summary results of the feature selection are collected in Table 4, where they are compared with earlier gathered results of experiments using the decision tree formalism (the *DTLevelImp* column) [11]. As it can be seen, Generational Feature Selection based on decision rules discover about 15 ÷ 16 truly relevant features. In turn, the decision tree based approach discovered all twenty relevant features without false positives. They didn't exceed the threshold of removing probability > 0.5.

Initial results are promising, however, it could be identified problem with the unequivocal definition of the threshold used to separate truly relevant feature from the other irrelevant. For example, in case of Madelon dataset, the *f5* feature which is random noise was discovered 2, 3 or 4 times during 10-fold cross-validation (see Table 4), thus their probability estimator for removing is even 0.6. Similar values of this estimator could be observed for known truly relevant features *f456* or *f154*. These features are weakly relevant and they are difficult to detect. Also, the influence of the process of features values discretization may be crucial for discovering them. The proposed algorithm of generational feature selection seems to be robust and let to find weakly relevant important attributes due to sequential elimination of strongly relevant attributes.

Table 4. The summary results gathered in **10-folds** using *RQI* based on *CN2* and *own rule quality measure* and the *WRACC* and *Laplace* evaluation functions. These results are compared with *DTLevelImp* algorithm. Bold names denote truly relevant features, other denotes irrelevant ones. The grey colored cells denote the feature set indicated as relevant. 1 - feature name, 2 - # of selections, 3 - removing probability

WRACC + CN2 RQI			WRACC + own RQI			Laplace			DTLevelImp		
1	2	3	1	2	3	1	2	3	1	2	3
f49	10	0.0	f49	10	0.0	f49	10	0.0	f29	10	0.0
f65	10	0.0	f65	10	0.0	f65	10	0.0	f49	10	0.0
f129	10	0.0	f129	10	0.0	f129	10	0.0	f65	7	0.3
f242	10	0.0	f242	10	0.0	f242	10	0.0	f106	10	0.0
f337	10	0.0	f337	10	0.0	f337	10	0.0	f129	10	0.0
f339	10	0.0	f339	10	0.0	f339	10	0.0	f154	10	0.0
f443	10	0.0	f443	10	0.0	f443	10	0.0	f242	10	0.0
f454	10	0.0	f454	10	0.0	f454	10	0.0	f282	10	0.0
f473	10	0.0	f473	10	0.0	f473	10	0.0	f319	10	0.0
f476	10	0.0	f476	10	0.0	f476	10	0.0	f337	10	0.0
f106	10	0.0	f106	10	0.0	f494	10	0.0	f339	10	0.0
f494	9	0.1	f494	9	0.1	f106	9	0.1	f379	10	0.0
f319	8	0.2	f379	8	0.2	f379	9	0.1	f434	8	0.2
f379	6	0.4	f319	7	0.3	f29	7	0.3	f443	10	0.0
f282	6	0.4	f282	7	0.3	f452	6	0.4	f452	10	0.0
f456	4	0.6	f456	5	0.5	f154	3	0.7	f454	6	0.4
f154	2	0.8	f154	4	0.6	f405	3	0.7	f456	5	0.5
f29	2	0.8	f29	2	0.8	f5	2	0.8	f473	10	0.0
f286	2	0.8	f286	2	0.8	f8	2	0.8	f476	10	0.0
f452	1	0.9	f5	2	0.8	f178	2	0.8	f494	6	0.4
f434	1	0.9	f190	2	0.8	f206	2	0.8	f5	4	0.6
f5	3	0.7	f452	1	0.9	f229	2	0.8	f177	1	0.9
f362	1	0.9	f434	1	0.9	f282	2	0.8	f359	1	0.9
f227	1	0.9	f362	1	0.9	f286	2	0.8	f414	1	0.9
f190	2	0.8	f227	1	0.9	f39	1	0.9	f85	1	0.9
f264	1	0.9	f264	1	0.9	f43	1	0.9	f256	1	0.9
f357	1	0.9	f357	1	0.9	f52	1	0.9	f112	1	0.9
f191	1	0.9	f432	1	0.9	f103	1	0.9	f286	2	0.8
f229	1	0.9	f266	1	0.9	f153	1	0.9	f216	1	0.9
f194	1	0.9	f287	1	0.9	f174	1	0.9	f292	2	0.8
f12	1	0.9	f236	1	0.9	f201	1	0.9	f343	1	0.9
f174	1	0.9	f269	1	0.9	f246	1	0.9	f74	1	0.9
f489	1	0.9				f279	1	0.9	f148	1	0.9
f244	1	0.9				f284	1	0.9	f472	1	0.9
f41	1	0.9				f304	1	0.9	f203	1	0.9
f350	1	0.9				f306	1	0.9	f211	1	0.9
						f319	1	0.9	f304	1	0.9
						f333	1	0.9	f7	1	0.9
						f334	1	0.9	f440	1	0.9
						f335	1	0.9	f323	1	0.9
						f357	1	0.9	f245	1	0.9
						f358	1	0.9			
						f381	1	0.9			
						f403	1	0.9			
						f411	1	0.9			

Acknowledgments. This work was supported by the Center for Innovation and Transfer of Natural Sciences and Engineering Knowledge at the University of Rzeszów.

References

1. Kuhn, M., Johnson, K.: Applied Predictive Modeling. Springer, New York (2013). <https://doi.org/10.1007/978-1-4614-6849-3>
2. Liu, H., et al.: Feature selection aspects. In: Liu, H., Motoda, H. (eds.) Feature Selection for Knowledge Discovery and Data Mining. The Springer International Series in Engineering and Computer Science, vol. 454, pp. 43–72. Springer, Boston (1998). https://doi.org/10.1007/978-1-4615-5689-3_3
3. Nilsson, R., Peña, J.M., Björkegren, J., Tegnér, J.: Detecting multivariate differentially expressed genes. BMC Bioinform. **8**, 150 (2007)
4. Paja, W., Wrzesień, M., Niemiec, R., Rudnicki, W.R.: Application of all-relevant feature selection for the failure analysis of parameter-induced simulation crashes in climate models. Geosci. Model Dev. **9**, 1065–1072 (2016)
5. Rudnicki, W.R., Wrzesień, M., Paja, W.: All relevant feature selection methods and applications. In: Stańczyk, U., Jain, L.C. (eds.) Feature Selection for Data and Pattern Recognition. SCI, vol. 584, pp. 11–28. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-45620-0_2
6. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Mach. Learn. **46**, 389–422 (2002)
7. Johannes, M., Brase, J.C., Frohlich, H., Gade, S., Gehrman, M., Falth, M., Sultmann, H., Beifbarth, T.: Integration of pathway knowledge into a reweighted recursive feature elimination approach for risk stratification of cancer patients. Bioinformatics **26**(17), 2136–2144 (2010)
8. Stoppiglia, H., Dreyfus, G., Dubois, R., Oussar, Y.: Ranking a random feature for variable and feature selection. J. Mach. Learn. Res. **3**, 1399–1414 (2003)
9. Tuv, E., Borisov, A., Torkkola, K.: Feature selection using ensemble based ranking against artificial contrasts. In: International Symposium on Neural Networks, pp. 2181–2186 (2006)
10. Paja, W.: Feature selection methods based on decision rule and tree models. In: Czarnowski, I., Caballero, A.M., Howlett, R.J., Jain, L.C. (eds.) Intelligent Decision Technologies 2016. SIST, vol. 57, pp. 63–70. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39627-9_6
11. Paja, W.: Generational feature elimination to find all relevant feature subset. In: Czarnowski, I., Howlett, R.J., Jain, L.C. (eds.) IDT 2017. SIST, vol. 72, pp. 140–148. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-59421-7_13
12. Guyon, I., Gunn, S., Ben-Hur, A., Dror, G.: Result analysis of the NIPS 2003: feature selection challenge. In: Advances in Neural Information Processing Systems, vol. 17, pp. 545–552 (2013)
13. Paja, W., Pancierz, K., Grochowalski, P.: Generational feature elimination and some other ranking feature selection methods. In: Stańczyk, U., Zielosko, B., Jain, L.C. (eds.) Advances in Feature Selection for Data and Pattern Recognition. ISRL, vol. 138, pp. 97–112. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-67588-6_6



A Partial Demand Fulfilling Capacity Constrained Clustering Algorithm to Static Bike Rebalancing Problem

Yi Tang and Bi-Ru Dai^(✉)

Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan, R.O.C.
ml0515038@mail.ntust.edu.tw, brdai@csie.ntust.edu.tw

Abstract. Nowadays, bike sharing systems have been widely used in major cities around the world. One of the major challenges of bike sharing systems is to rebalance the number of bikes for each station such that user demands can be satisfied as much as possible. To execute rebalancing operations, operators usually have a fleet of vehicles to be routed through stations. When rebalancing operations are executing at nighttime, user demands usually are small enough to be ignored and this is regarded as the static bike rebalancing problem. In this paper, we propose a Partial Demand Fulfilling Capacity Constrained Clustering (PDF3C) algorithm to reduce the problem scale of the static bike rebalancing problem. The proposed PDF3C algorithm can discover outlier stations and group remaining stations into several clusters where stations having large demands can be included by different clusters. Finally, the clustering result will be applied to multi-vehicle route optimization. Experiment results verified that our PDF3C algorithm outperforms existing methods.

Keywords: Bike rebalancing · Clustering · Mixed integer linear programming

1 Introduction

Nowadays, as the first-and-last mile connections, the bike sharing systems have been widely used in major cities around the world, and there are over 1400 systems online [9]. The most convenient feature of bike sharing systems is that a user can borrow a bike in any station and then return it to any other station.

However, user demands for different stations are usually unbalanced such that some stations are short of bikes to be borrowed and some stations do not have enough docks to be hooked. With unbalanced stations, fewer users can be served and the revenue of operators will be reduced. Therefore, operators usually have a fleet of trucks to rebalance the number of bikes between stations. To determine the rebalancing route and rebalancing operations for each truck is the bike rebalancing problem.

Since the traffic congestion and parking locations will not be a problem while the rebalancing fleet travels in the city during the night [11], we focus on the static bike rebalancing problem which assumes the rebalancing operations are executing at nighttime while the user operations at this time are usually small enough to be ignored.

In general, the static bike rebalancing problem can be treated as a One-commodity Capacitated Pickup and Delivery Problem [2] which is an NP-hard problem and will be

hard to find the optimal solution in limited time when the problem scale increases. For a 200 station case with 3 trucks, even the advanced optimization model [11] may take several hours to find the optimal solution. While finding out the optimal solution becomes difficult, several works have been tried to find a good enough near-optimal solution during the limited time, where [1] proposed an approximation algorithm, [10, 13, 15] developed different heuristic methods, [2, 4, 6] applied the meta-heuristic techniques and [3, 7, 8, 14] used the clustering techniques to divide the multi-vehicle static bike rebalancing problem into several single-vehicle static bike rebalancing problem to reduce the problem scale.

To further reduce the problem scale, Liu et al. [8] considered outlier stations whose demands are large and hard to be satisfied. However, their method still cannot deal with stations whose demand is larger than the vehicle capacity directly.

In this paper, to deal with stations with large demands and utilize the outlier station discovering to further reduce the problem scale of the static bike rebalancing problem, based on the CCKC algorithm [8], we propose a Partial Demand Fulfilling Capacity Constrained Clustering (PDF3C) algorithm which allows the demands of one station to be partially considered by different clusters and utilizes the average saved shortage to discriminate the considering priority of stations. Experimental results show that for the large-scale static bike rebalancing problem with some stations having large demands, the proposed PDF3C clustering method can get better performance than existing methods.

The rest sections of this paper are organized as follows. We summarize the strategies of existing methods to the static bike rebalancing problem in Sect. 2. The problem definitions are demonstrated in Sect. 3. The proposed method is presented in Sect. 4. Experiment results are presented in Sect. 5. Finally, the conclusion of this paper is in Sect. 6.

2 Related Works

In this section, we will briefly summarize the strategies of existing methods and introduce several clustering methods with their reduction to the multi-vehicle static bike rebalancing problem.

When traditional methods cannot get optimal or good enough near-optimal solutions in limited time, some methods were proposed to get better solutions, including heuristic methods [13, 15], meta-heuristic methods [2, 4, 6] and clustering methods [3, 7, 8, 14]. The strategies for these methods are different. The heuristic methods usually give some rules related to the problem to guide the searching or construction of the solution, the meta-heuristic methods apply their own mechanism to utilize the searching experience in solution space to improve the searching process and the clustering methods try to reduce the solution space and keep the solution quality simultaneously. Some of these methods are combined with each other or with other methods to become hybrid methods.

Clustering methods can be used to reduce the problem scale of the multi-vehicle bike rebalancing problem. Schuijbroek et al. [14] proposed a Cluster-First Route-Second approach to divide the multi-vehicle bike rebalancing problem into single-vehicle bike rebalancing problems, where each cluster represents a vehicle, and

the travel distance for each cluster is approximated by the Maximum Spanning Star. Forma et al. [3] proposed a 3-step Math Heuristic method. In addition to divide the original problem into several single-vehicle bike rebalancing problems, the travel routes between clusters are determined by Step 2 in their method and the decision variables in the MILP model are reduced according to the travel routes.

To further reduce the problem scale, Liu et al. [8] proposed a Capacity Constrained K-centers Clustering method which use the balance condition to discover outlier stations before solving the bike rebalancing problems. However, this method is not able to deal with outlier stations with the number of rebalancing operations being larger than the vehicle capacity.

3 Problem Formulation

In this section, we will introduce the static bike rebalancing problem and the station clustering problem.

The static bike rebalancing problem is to determine the rebalancing route and rebalancing instructions for each vehicle such that the expected shortage of each station in the next day will be as low as possible. In this paper, the target inventory for each station is determined by the penalty function provided in [12]. Assume the penalty function is given then the static bike rebalancing problem is defined as follows.

Definition 1: Static Bike Rebalancing Problem

Given one depot and a set of stations with their initial bikes or vacancies, the penalty function for each station, the travel cost between stations and depot, the load and unload time for single rebalancing instruction, the total time budget for rebalancing operations and a weight alpha to tradeoff between travel cost and rebalancing operations, the static bike rebalancing problem is to determine the rebalancing route and rebalancing instructions for each vehicle that minimize the total travel cost and the shortage for each station.

In addition, in order to simplify the original problem and satisfy some stations with large demands, each vehicle is restricted to visit each station at most once but each station can be visited by more than one vehicle.

Usually, after determining the final bike inventory for each station at the end of a day, there are only a few hours remained for solving the static bike rebalancing problem and conducting the rebalancing operations. For the small-scale static bike rebalancing problem, several works [7] have solved it by mixed integer linear programming (MILP) methods. However, for a large-scale static bike rebalancing problem, pure MILP methods are often not able to get the optimal or a good enough near-optimal solution in limited time.

To reduce the problem scale, clustering is a good technique to allocate stations for each vehicle. This is called station clustering problem and is defined as follows.

Definition 2: Station Clustering Problem

Given a static bike rebalancing problem, the station clustering problem is to allocate stations into clusters, where each cluster represents one vehicle, according to the travel cost and the rebalancing operations to reduce the problem scale of the original problem.

Note that some stations are probably not assigned to any cluster and will be regarded as outlier stations.

After problem definitions, the proposed method to solve the identified problems will be introduced in the next section.

4 Proposed Method

In this section, we will first give an overview of our framework and then demonstrate the proposed method in two parts, where the target inventory determination part is in Sect. 4.2 and the rebalancing route optimization part is in Sect. 4.3.

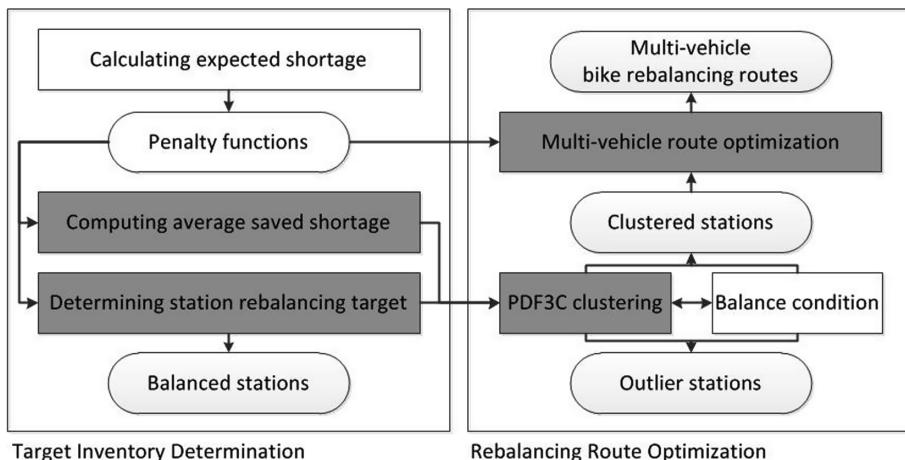


Fig. 1. Proposed framework

4.1 Framework

The whole framework can be divided into two parts, one is the target inventory determination part and the other is the rebalancing route optimization part. The framework is illustrated in Fig. 1, and components filled by gray color are designed or revised by this paper. In the target inventory determination part, we will need to calculate the expected shortage for each station first. This process can be done by different methods, such as some stochastic analysis or demand prediction methods. In this paper, our expected shortage is calculated by the method proposed in [12] and is represented as the penalty function for each station. In order to discriminate the considering priority for each station and discover outlier stations according to the vehicle capacity, the average saved shortage and the station rebalancing target are computed, respectively. During this process, stations with zero rebalancing targets will be regarded as balanced stations and will not be taken into the rebalancing problem. Next, in the rebalancing route optimization part, according to the station rebalancing target and the average saved shortage, we design a clustering algorithm, named PDF3C, to generate clustered stations for each

vehicle and discover some outlier stations according to the balance condition. Finally, the penalty function and clustered stations will be taken as inputs of the MILP model to obtain the optimized bike rebalancing routes and rebalancing instructions.

4.2 Target Inventory Determination

In the target inventory determination part, we use the same penalty function as [12] to represent the expected shortage for each station. Then the station rebalancing target can be calculated accordingly. Different from [8], we further define the average saved shortage to determine the priority of stations, as described below.

Penalty Function. The penalty function represents the expected shortage number of docks or bikes incurred by users during next day. For each station i having capacity c_i , with a number of bikes s_i after the rebalancing operations, the penalty function $f_i(s_i)$ is defined discretely [12]:

$$shortage = f_i(s_i) \quad s_i = 0, \dots, c_i \quad (1)$$

Station Rebalancing Target. The station rebalancing target is the number of bikes required to be load or unload if we want to get the minimum shortage for a bike station. Given the penalty function of station i , since the convexity of penalty function, there exists a number of bikes s_i^* corresponding to the minimum shortage. Assume that the number of bikes before rebalancing operations is s_i^0 , the station rebalancing target b_i is defined as:

$$b_i = s_i^* - s_i^0 \quad (2)$$

Average Saved Shortage. When the rebalancing target of some stations cannot be satisfied, it is necessary to determine which station will have higher priority and need to be satisfied first. Therefore, we define the average saved shortage for each station i as follows:

$$ass_i = \left| \frac{s_i^* - s_i^0}{b_i} \right| \quad (3)$$

Stations with larger average saved shortages will be given higher priority because for such stations, more shortages can be saved by a single rebalancing instruction in average.

In the next part, the station rebalancing target and the average saved shortage for each station will be used to generate station clusters. Then, the multi-vehicle route optimization will be performed based on the penalty function and station clusters.

4.3 Rebalancing Route Optimization

The linear programming model is a common way to solve the static bike rebalancing problem. However, for the large-scale static bike rebalancing problem, even the advanced MILP model [11] will take a very long time to get a good enough solution. In order to reduce the problem scale, we propose a clustering algorithm, named Partial Demand Fulfilling Capacity Constrained Clustering (PDF3C) algorithm, to allow the

demands of one station to be partially considered by different clusters. The same balance condition mentioned in [8] will be used in this paper and is stated below.

Balance Condition. Given the vehicle capacity of k and a set of stations belonging to cluster I , where $B(I)$ is the absolute value of the sum of rebalancing targets for all stations in cluster I , the balance condition is defined as follows:

$$B(I) \leq k \quad \text{where } B(I) = \left| \sum_{i \in I} b_i \right| \quad (4)$$

If the balance condition is not violated, the rebalancing target for all stations in the same cluster will be fully satisfied in one route using one vehicle. Note that one route represents a vehicle starting from the depot with its initial bikes or vacancies, passing through several stations, and finally going back to the depot.

Partial Demand Fulfilling Capacity Constrained Clustering

The difficulty of using MILP model to solve the static bike rebalancing problem mainly depends on the number of decision variables and constraints [3]. Although there are several works applying clustering techniques to divide the multi-vehicle static bike rebalancing problem into several single-vehicle bike rebalancing problems to reduce the problem scale [3, 7, 8, 14], only [8] has considered outlier stations, whose rebalancing targets cannot be fulfilled completely, to further reduce the problem scale to get better effectiveness.

Liu et al. [8] proposed a clustering algorithm named Capacity Constrained K-centers Clustering (CCKC) to discovery outlier stations by the capacity balanced condition mentioned above. However, CCKC does not consider the following two points: 1. to deal with stations whose rebalancing targets are larger than the vehicle capacity; 2. to discriminate important stations when the saved shortage for each station by one instruction is heterogeneous.

The importance of considering point 1 is that for those stations whose rebalancing targets are larger than the vehicle capacity, the saved shortage will usually be large and have a serious impact on the revenue of operators. The consideration of point 2 is trying to allow each rebalancing operation and each bike in the system to be utilized to save the bike shortage as much as possible. When the saved shortage for a station by one instruction is heterogeneous, we should satisfy those stations with higher saved shortages first.

In order to overcome these two points, in this paper, we propose the Partial Demand Fulfilling Capacity Constrained Clustering (PDF3C) algorithm which allows the rebalancing target to be partially considered and takes into account the saved shortage for different stations using average saved shortage. The flowchart and the pseudo code of the proposed algorithm are illustrated in Fig. 2 and Algorithm 1, respectively.

Before the details of the proposed algorithm, we demonstrate the flowchart first. The initialization step is to clear the cluster assignment for each station and reset the station set. After that, we then mark up stations whose rebalancing targets are larger than the vehicle capacity and assign each of the remaining stations to its closest cluster. In the following remove station step, while the balance condition for any one cluster is

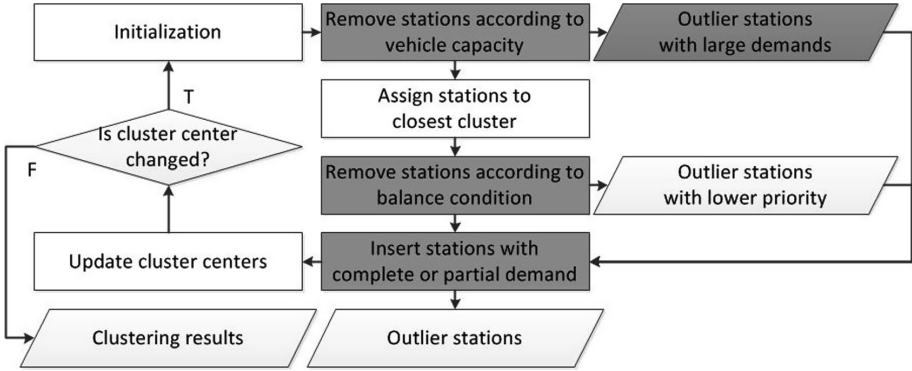


Fig. 2. Partial demand fulfilling capacity constrained clustering algorithm flowchart

violated, we keep removing some stations with lower priority until the balance condition is not violated for that cluster. Next, in the insert station step, we try to utilize the remaining capacity for each vehicle to cover outlier stations. After the insert station step, stations still belonging to no cluster are real outlier stations. Finally, cluster centers will be updated and all steps will be repeated until centers are not changed.

Algorithm 1 *PDF3C*($\mathbf{TC}, \mathbf{b}, k, C, N_s, N_{ori}, E, \varepsilon$)

Input: $TC_{ij}, b_i, k, C, N_s, N_{ori}, E, \varepsilon$;

Output: C ;

- 1: $N_s \leftarrow N_{ori}$;
 - 2: **for** $i \in N_s$ **do** $c(i) = 0$;
 - 3: **for** $b_i > k, i \in N_s$ **do** $c(i) = -2$;
 - 4: **for** $c(i) = 0, i \in N_s$ **do** $c(i) = \arg \min_{j \in E} (TC_{ij})$;
 - 5: **for** $B(I) > k, I \in C$ **do**
 - 6: **while** $B(I) > k$ **do**
 - 7: $q = \arg \min_{i \in I^*} (\sum_{j \in E} TC_{ij} \cdot ass_i \cdot |b_i|)$, where $I^* = \{i \mid B(I \setminus i) < B(I)\}$
 - 8: $c(q) = -1, I = I \setminus q$;
 - 9: **find** $l = \arg \max_{i, c(i) = -1, -2} (\sum_{j \in E} TC_{ij} \cdot ass_i \cdot |b_i|)$
 - 10: create $a = l$;
 - 11: **if** $\exists CC = \{I \mid B(I \cup a) < k, \frac{ass_i \cdot |ab_{ll}|}{\min_{j \in I} (TC_{ij})} > \varepsilon\}$ **then**
 - 12: **if** $b_l = ab_{ll}$ **then**
 - 13: delete $a, c(l) = E(I), I = \arg \min_{Q \in CC} (\min_{j \in Q} (TC_{ij}))$
 - 14: **else**
 - 15: add a to $N_s, c(a) = E(I), I = \arg \min_{Q \in CC} (\min_{j \in Q} (TC_{ij}))$
 - 16: $b_l = b_l - ab_{ll}, b_a = ab_{ll}$;
 - 17: **for** $c(i) = -3, -4, i \in N_s$ **do** $c(i) = c(i) + 2$;
 - 18: **else**
 - 19: delete $a, c(l) = c(l) - 2$;
 - 20: **go to** step 9 **till** $\nexists i$ that $c(i) = -1, -2$;
 - 21: **for** $I \in C$ **do** $E'(I) = \arg \min_{i, c(i) = E(I)} (\sum_{j, c(j) = E(I)} TC_{ij})$;
 - 22: **if** $E' \neq E$ **then** $E = E'$, **go to** step 1 **else** **go to** step 23;
 - 23: **return** clustering result C .
-

We now begin to demonstrate the proposed algorithm. Inputs of Algorithm 1 include the travel cost \mathbf{TC} , rebalancing targets \mathbf{b} , vehicle capacity k , set of clusters C , set of current station nodes N_s , set of original station nodes N_{ori} , set of initial cluster centers E , which can be generated randomly or by some other methods, and the benefit threshold ε . First, in the initialization step, Step 1 resets current station set to the original station set and Step 2 clears the cluster assignments for each station, and then Step 3 markups outlier stations with large rebalancing targets. Step 4 assigns each of the remaining station to its closest cluster. When the balance condition is violated, Step 5–8 keep removing stations with lower priority which is determined by the distance to other clusters and the average saved shortage for that station. Next, in Step 9–20, stations without a cluster assignment are reassigned where the priority of these stations are similar to Step 5–8 but in a contrary way. For each reassigned station, the available candidate cluster set is determined by the benefit which is computed by the average saved shortage and the available rebalancing target where the available rebalancing target is the maximum rebalancing target that can be included into the current considered cluster. If the available rebalancing target is equal to the original rebalancing target, it means that the rebalancing target of the reassigned station can be fully included by the candidate cluster. Therefore, the reassigned station will be assigned to the candidate cluster directly. Otherwise, the reassigned station will be duplicated with the available rebalancing target and the duplicated one will be assigned to the candidate cluster while the rebalancing target of the original reassigned station will be subtracted by the available rebalancing target. In other words, when the available rebalancing target is not equal to the original rebalancing target, we will split the reassigned station into two stations with the available rebalancing target and the remaining rebalancing target respectively. After adjusting the clustering result with considering the balance condition, cluster centers are updated in Step 21. Then, Step 22 determines whether these cluster centers are changed. If not, Step 23 returns the clustering result; otherwise, the algorithm restarts from Step 1 with these updated cluster centers.

In above, we have demonstrated the PDF3C algorithm completely. Since the proposed algorithm is based on the CCKC algorithm, we summarize main differences as follows.

- First, in Step 3, stations with too large rebalancing target are excluded before station assignments. By this way, we avoid the problem occurred in [5] that all stations will be removed from a cluster if the cluster includes a station whose rebalancing target is larger than the vehicle capacity.
- Second, in Step 11–20, to deal with stations with large rebalancing target and fully utilize the capacity of each vehicle, the rebalancing target of a station is allowed to be partially considered by different clusters.
- Third, in Step 11, in addition to the original distance threshold in CCKC algorithm, we also take into account how much shortage can be saved when inserting a station into a cluster to further identify valuable stations to be included.
- Forth, in Step 7 and Step 9, because the penalty function is different for each station, we include the concept of average saved shortage to discriminate which station can save more shortage.

Table 1. The summary of notations

Sets and parameters	Description
V	The vehicle set, indexed by $v = 1, \dots, V $
k_v	Capacity of vehicle $v \in V$
N_s	The station node set, indexed by $i = 1, \dots, N_s $
N_0	The station node set with depot where the depot is indexed by $i = 0$
s_i^0	Number of bikes at station i before the rebalancing operation
c_i	Number of docks at station i
$f_i(s_i)$	The penalty function for station $i \in N_s$
t_{ij}	Travel cost from station i to station j here is the travel time in seconds
L	Time for remove a bike from a dock and load it onto the vehicle
U	Time for unload a bike from the vehicle and hook it to a dock
T	Total time for the rebalancing operation
α	Trade-off factor of travel cost and bike shortage
Decision variables	Description
x_{ijv}	Binary variable equals to 1 if vehicle travels from station i to station j , equals to 0 for otherwise
y_{ijv}	Number of bikes on vehicle when it travels from station i to station j , equals to 0 if vehicle dose not travel from station i to station j
y_{iv}^L	Number of bikes loaded onto vehicle v at station i
y_{iv}^U	Number of bikes unloaded from vehicle v at station i
s_i	Number of bikes at station i after rebalancing operation

After the PDF3C algorithm, stations whose rebalancing target is not zero will be divided into clusters and some of them will become outlier stations. Next, in the multi-vehicle route optimization step, the MILP model will use the clustering result to reduce the problem scale.

Multi-vehicle Route Optimization

In this step, we use the Arc Index formulation which is the MILP model proposed by [11] to solve the static bike rebalancing problem. To reduce the problem scale for the MILP model, the creation of decision variables and constraints, which are referenced by vehicles and stations, will be decided according the clustering result. In order to see how many decision variables and constraints can be reduced, we then briefly introduce the Arc Index Formulation where the notations used to describe the MILP model are summarized in Table 1.

Because of the limited space, some details such as the binary and general integrality constraints, non-negativity constraints, sub-tour elimination constraints, penalty function piecewise constraints and the decision variables relative to those constraints, which can be found in [11], will not be introduced here.

The Arc Index (AI) is formulated by following objective function and constraints:

$$\min \sum_{i \in N_s} f_i(s_i) + \alpha \sum_{i \in N_0} \sum_{j \in N_0} \sum_{v \in V} t_{ij} x_{ijv} \quad (5)$$

s.t.

$$s_i = s_i^0 - \sum_{v \in V} (y_{iv}^L - y_{iv}^U) \quad \forall i \in N_0 \quad (6)$$

$$y_{iv}^L - y_{iv}^U = \sum_{j \in N_0, i \neq j} y_{ijv} - \sum_{j \in N_0, i \neq j} y_{jiv} \quad \forall i \in N_0, \forall v \in V \quad (7)$$

$$y_{ijv} \leq k_v x_{ijv} \quad \forall i, j \in N_0, i \neq j, \forall v \in V \quad (8)$$

$$\sum_{j \in N_0, i \neq j} x_{ijv} = \sum_{j \in N_0, i \neq j} x_{jiv} \quad \forall i \in N_0, \forall v \in V \quad (9)$$

$$\sum_{j \in N_0, i \neq j} x_{ijv} \leq 1 \quad \forall i \in N_s, \forall v \in V \quad (10)$$

$$\sum_{j \in N_0, i \neq j} x_{ijv} \leq 1 \quad \forall i \in N_0, \forall v \in V \quad (11)$$

$$\sum_{v \in V} y_{iv}^L \leq s_i^0 \quad \forall i \in N_0 \quad (12)$$

$$\sum_{v \in V} y_{iv}^U \leq c_i - s_i^0 \quad \forall i \in N_0 \quad (13)$$

$$\sum_{i \in N_0} (y_{iv}^L - y_{iv}^U) = 0 \quad \forall v \in V \quad (14)$$

$$\sum_{i \in N_s} (Ly_{iv}^L + Uy_{iv}^U) + \sum_{i \in N_s} (Ly_{0iv}^L + Uy_{0iv}^U) + \sum_{i, j \in N_0, i \neq j} t_{ij} x_{ijv} \leq T \quad \forall v \in V \quad (15)$$

The objective function (5) minimizes the bike shortage and the travel cost with a trade-off parameter α . Constraints (6) are inventory-balance constraints. For each station, the number of bikes after rebalancing operation is the original number of bikes plus the sum of load (positive) and unload (negative) instructions by all vehicles. Constraints (7) represent the conservation of inventory for each vehicle. When a vehicle travels through one station, checking the cross in y_v can determine the past station and the next station, then the difference of two numbers in y_v is the number of bikes loaded or unloaded in that station. Constraints (8) limit the vehicle capacity. If a vehicle does not travel through two stations directly, the corresponding element in x_v will be zero, hence there is no capacity. Constraints (9) ensure the travel frequency from one station is equal to the travel frequency to that station. Constraints (10) restrict each station to be visited

at most once by the same vehicle, while the depot can be visited more than one time. To fit our problem definition, we modify constraints (10) to constraints (11) which also restrict that the depot can be visited at most one time. Constraints (12) and constraints (13) limit the pick-up quantity and drop-off quantity for one station by all vehicles respectively. Constraints (14) make sure that the sum of pick-up quantity and the sum of drop-off quantity are equal. Constraints (15) are the time limit constraints for each vehicle which consist of the total instruction time and the total travel time.

In addition, since the execution time allowed for the route optimization is limited, time assignments for each single-vehicle bike rebalancing problem will be difficult because some clusters with more stations will take a longer time. To address the problem of time assignments, we use only one MILP model to solve the multi-vehicle bike rebalancing problem while the decision variables and constraints are still can be reduced according to the clustering result.

After reducing the decision variables and constraints of MILP model, we conduct the optimization to get the final result of the static bike rebalancing problem.

So far, we have already demonstrated the proposed method completely. The proposed method starts from using the penalty function to compute the station rebalancing target and the average saved shortage. The average saved shortage is used to discriminate the importance of each station and the station rebalancing target is used to check the balance condition. Then, before using the MILP model to solve the static bike rebalancing problem, we first utilize the clustering algorithm to divide all stations into several clusters and some outlier stations and then use these assignments to reduce the number of decision variables and constraints in the MILP model. Finally, the MILP model with reduced decision variables and constraints is used to solve the final rebalancing routes and instructions for each vehicle in limited time.

In summary, based on the clustering result generated by the proposed PDF3C algorithm, which allows the station rebalancing target to be partially considered by different vehicles, stations with large rebalancing targets can be satisfied by multiple vehicles. In addition, some outlier stations can be further fulfilled according to the priority provided by the designed average saved shortage.

5 Experiment

In this section, we will introduce the dataset, the baseline method and competitors in Sect. 5.1, and presents the experiment results in Sect. 5.2.

5.1 Comparative Environment

We used the dataset provided by [3]. This dataset has 200 stations with different workloads corresponding to different initial inventories according to the light, real and heavy case for each station. There are 3 vehicles with the capacity of 25 in our rebalance planning. The pick-up and drop-off time for each rebalancing instruction is 60 s and the time budget for rebalancing operations is 18000 s.

To verify the effectiveness of the proposed PDF3C algorithm, denoted as PDF3C, we use the pure MILP model named Arc Index Formulation as the baseline method

Table 2. Experiment results for the average case and the best case

Instance	PDF3C		CCKC		3-Step MH		Arc Index	
	Avg. Obj.	Avg. Gap.	Avg. Obj.	Avg. Gap.	Avg. Obj.	Avg. Gap.	Obj.	Gap.
light 75	509.555	0.399	508.463	0.418	512.222	0.010	520.528	5.194
light 100	685.204	0.274	687.854	0.035	688.304	0.010	722.039	8.602
light 125	855.062	0.205	855.866	0.250	861.303	0.010	924.101	10.796
light 150	1039.630	0.632	1043.454	0.765	1052.244	0.459	1127.402	11.458
light 175	1233.580	1.362	1233.324	1.238	1250.976	1.494	1333.669	12.234
light 200	1444.372	1.828	1455.367	2.244	1514.556	0.611	1548.963	12.848
real 75	507.252	0.109	504.384	0.019	504.724	0.004	516.668	4.733
real 100	668.604	0.146	667.845	0.082	668.952	0.010	681.892	4.270
real 125	834.725	0.296	837.647	0.420	841.998	0.410	910.069	10.882
real 150	1018.724	0.223	1040.734	0.409	1030.525	0.771	1124.391	12.260
real 175	1206.169	1.211	1227.719	1.269	1249.441	0.879	1329.185	12.779
real 200	1428.183	3.233	1444.278	2.984	1480.173	0.982	1541.856	13.497
heavy 75	561.730	0.911	578.947	0.793	555.386	1.249	593.341	12.960
heavy 100	780.531	2.441	804.379	1.333	767.020	1.242	951.194	26.050
heavy 125	1023.788	3.577	1035.414	2.286	1022.520	2.196	1282.824	28.981
heavy 150	1302.080	5.719	1315.156	3.890	1336.187	2.697	1610.413	28.884
heavy 175	1637.992	7.923	1615.940	2.596	1632.715	2.370	1921.630	26.361
heavy 200	1991.767	5.803	1995.559	2.573	1973.379	2.319	2289.897	22.679
Instance	PDF3C		CCKC		3-Step MH		Arc Index	
	Best Obj.	Gap.	Best Obj.	Gap.	Best Obj.	Gap.	Obj.	Gap.
light 75	509.545	0.397	508.434	0.407	512.217	0.010	520.528	5.194
light 100	684.942	0.270	687.840	0.021	688.304	0.010	722.039	8.602
light 125	854.803	0.179	855.430	0.177	861.303	0.010	924.101	10.796
light 150	1038.536	0.527	1041.374	0.558	1051.487	0.347	1127.402	11.458
light 175	1227.199	0.854	1228.484	0.839	1248.653	1.298	1333.669	12.234
light 200	1441.197	1.686	1446.067	1.549	1513.980	0.573	1548.963	12.848
real 75	507.252	0.088	504.384	0.010	504.724	0.000	516.668	4.733
real 100	668.597	0.119	667.845	0.076	668.952	0.010	681.892	4.270
real 125	834.330	0.245	837.313	0.381	841.496	0.342	910.069	10.882
real 150	1018.612	0.209	1039.471	0.287	1030.022	0.737	1124.391	12.260
real 175	1199.583	0.639	1219.084	0.556	1249.082	0.852	1329.185	12.779
real 200	1411.006	1.776	1424.283	1.687	1477.667	0.834	1541.856	13.497
heavy 75	561.382	0.804	578.746	0.749	554.514	1.036	593.341	12.960
heavy 100	777.920	2.140	801.396	0.900	765.693	1.123	951.194	26.050
heavy 125	1014.320	2.864	1028.533	1.610	1016.794	1.657	1282.824	28.981
heavy 150	1262.569	2.719	1304.244	2.971	1333.634	2.499	1610.413	28.884
heavy 175	1607.221	5.917	1606.779	1.951	1631.089	2.402	1921.630	26.361
heavy 200	1939.325	3.339	1977.846	1.748	1965.346	1.860	2289.897	22.679

[11], denoted as Arc Index. In addition to the baseline method, we have two other clustering methods as competitors, one is the original CCKC algorithm [8], denoted as CCKC, and the other is the 3-Step Math Heuristic method [3], denoted as 3-Step MH. All algorithms are adjusted based on the same assumption that the depot can be visited by each vehicle at most once which are similar as the constraint (10) and (11), and outlier station removing is also applied to CCKC to avoid cluster disappearing.

About the parameter setting, for the MILP model, the trade-off factor is 1/900 and the execution time limit is 3600 s [3]; for clustering algorithms, according to the trade-off factor, the benefit threshold of PDF3C is also 1/900, the distance threshold of CCKC is 900 and the diameter of 3-Step MH is 800 [3] where this value of diameter is set according to some empirical tuning. We also conducted some experiments to tune the parameter and got a same conclusion of the diameter setting.

5.2 Experiment Results

In our experiments, we compare the objective value and the optimality gap calculated by a commercial solver where the objective value demonstrates the goodness of founded solution and the optimality gap roughly represents how much can the founded solution be improved. In addition, since with different order of the same clustering result to reduce the problem scale of the MILP model will get different results, we observe experiment results both from the average case and the best case for all possible permutations of the clustering result.

All experiments were conducted on an Intel personal computer with Intel(R) Core (TM) i5-3470 CPU, 3.20 GHz and 8 GB memory running Microsoft Windows 7 Ultimate system where the MILP models were solved by the Gurobi optimizer [5] with version 7.0 in java code.

Experiment results for the average case and the best case are summarized in Table 2 and the best result for each instance is represented in bold font. In this table, instance names are represented by workload and the number of stations, Obj. represents the objective value of founded solution and Gap. represents the optimality gap of the founded solution in percentage which is calculated by the Gurobi optimizer.

As we can see, the proposed PDF3C algorithm outperforms two competitors and the baseline method in most instances, especially in the best case. As for the heavy workload instances in the average case we do not get the best result since the problem scale is still large because too many stations with partial demand are considered. However, the best case can be interpreted as the quality of clustering result. Our PDF3C algorithm utilizes the mechanism of partial demand including and the concept of average saved shortage to cover more valuable stations and generate clusters with higher potential quality.

6 Conclusions and Future Work

In this paper, we proposed a Partial Demand Fulfilling Capacity Constrained Clustering (PDF3C) algorithm to reduce the problem scale of the static bike rebalancing problem. The PDF3C algorithm can discover outlier stations and group remaining stations into several clusters. Furthermore, our PDF3C algorithm allows the demands of one station to be partially considered by different clusters and considers the average saved shortage for each station to increase the potential quality of generated clusters. Experiment results verified that using the clustering results generated by our algorithm to reduce the number of decision variables and constraints in the MILP model can get better results than other clustering algorithms or pure MILP model. In future works, based on current clustering results generated by our PDF3C algorithm, we will try to utilize other reduction manners to further reduce the problem scale to get better results.

References

1. Benchimol, M., Benchimol, P., Chappert, B., De La Taille, A., Laroche, F., Meunier, F., Robinet, L.: Balancing the stations of a self service “bike hire” system. *RAIRO-Oper. Res.* **45**(1), 37–61 (2011)
2. Chemla, D., Meunier, F., Calvo, R.W.: Bike sharing systems: solving the static re-balancing problem. *Discret. Optim.* **10**(2), 120–149 (2013)
3. Forma, I.A., Raviv, T., Tzur, M.: A 3-step math heuristic for the static repositioning problem in bike-sharing systems. *Transp. Res. Part B-Method.* **71**(Suppl. C), 230–247 (2015)
4. Di Gaspero, L., Rendl, A., Urli, T.: Constraint-based approaches for balancing bike sharing systems. In: Schulte, C. (ed.) *CP 2013. LNCS*, vol. 8124, pp. 758–773. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40627-0_56
5. Gurobi Optimization, Inc.: Gurobi Optimizer Reference Manual. <http://www.gurobi.com> (2016)
6. Ho, S.C., Szeto, W.Y.: Solving a static repositioning problem in bike-sharing systems using iterated tabu search. *Transp. Res. Part E Logist. Transp. Rev.* **69**(Suppl. C), 180–198 (2014)
7. Kloimüller, C., Papazek, P., Hu, B., Raidl, G.R.: A cluster-first route-second approach for balancing bicycle sharing systems. In: Moreno-Díaz, R., Pichler, F., Quesada-Arencibia, A. (eds.) *EUROCAST 2015. LNCS*, vol. 9520, pp. 439–446. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27340-2_55
8. Liu, J., Sun, L., Chen, W., Xiong, H.: Rebalancing bike sharing systems: a multi-source data smart optimization. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1005–1014. ACM (2016)
9. Meddin, R., DeMaio, P.: The bike sharing world map (2017). <http://www.metrobike.net>
10. Papazek, P., Kloimüller, C., Hu, B., Raidl, G.R.: Balancing bicycle sharing systems: an analysis of path relinking and recombination within a GRASP hybrid. In: Bartz-Beielstein, T., Branke, J., Filipič, B., Smith, J. (eds.) *PPSN 2014. LNCS*, vol. 8672, pp. 792–801. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10762-2_78
11. Raviv, T., Tzur, M., Forma, I.A.: Static repositioning in a bike-sharing system: models and solution approaches. *EJTL* **2**(3), 187–229 (2013)
12. Raviv, T., Kolka, O.: Optimal inventory management of a bike-sharing station. *IIE Trans.* **45** (10), 1077–1093 (2013)
13. Rainer-Harbach, M., Papazek, P., Raidl, G.R., Hu, B., Kloimüller, C.: PILOT, GRASP, and VNS approaches for the static balancing of bicycle sharing systems. *J. Glob. Optim.* **63**(3), 597–629 (2015)
14. Schuijbroek, J., Hampshire, R.C., van Hoeve, W.J.: Inventory rebalancing and vehicle routing in bike sharing systems. *EJOR* **257**(3), 992–1004 (2017)
15. Szeto, W.Y., Liu, Y., Ho, S.C.: Chemical reaction optimization for solving a static bike repositioning problem. *Transp. Res. Part D Transp. Environ.* **47**(Suppl. C), 104–135 (2016)



Detection of IP Gangs: Strategically Organized Bots

Tianyue Zhao^{1,3(✉)} and Xiaofeng Qiu²

¹ Henry M. Gunn High School, Palo Alto, CA 94306, USA
zhaotianyue@outlook.com

² Beijing University of Posts and Telecommunications, Beijing, China

³ NSFOCUS Inc., Santa Clara, CA 95054, USA

Abstract. Botnets, groups of malware-infected computers (bots) that perform cybersecurity attacks on the Internet, pose one of the most serious cybersecurity threats to many industries, including smart infrastructure [9, 10], Internet based companies, [11] and Internet of Things (IoT) [8]. There are many unconventional methods of organizing bots that are potentially advantageous to attackers. “Botnet”, as a technical term, cannot effectively describe these methods. With the vast amounts of Internet traffic data collected by security appliances, it is possible to reveal novel behavior of bots using data analysis algorithms. In this paper, we propose a concept called IP Gang to describe groups of bots from the perspective of the attacker’s business – we define IP Gangs to be groups of bots that often perform attacks together during a period of time. Crucially, we developed a fast, high-compatibility detection algorithm that can be deployed in wide-scale, industrial applications to effectively defend against IP Gangs. The detection algorithm is inspired by single-linkage clustering and optimized for large quantities of data. A test on a month (1.5 GB) of real life DDoS log data detected 21 IP Gangs, with 13916 bots in total. To analyze the behavior of the Gangs, we visualized the activity of each Gang with diagrams named “attack fingerprints” and confirmed that 15 of the detected Gangs displayed behavior that the concept of “botnet” alone cannot describe.

Keywords: IP gang · Botnet · Cybersecurity · Big data

1 Introduction

Botnets, groups of malware-infected computers (bots) that perform cybersecurity attacks on the Internet, pose one of the most serious cybersecurity threats to many industries. Smart infrastructure such as power grids have been attacked to deny hundreds of thousands of people basic services [9, 10]. Internet-based industries have been hit with massive Distributed Denial of Service (DDoS) attacks that can render large websites inoperative for entire hours [11]. Internet of Things (IoT) devices such as webcams are regularly hijacked to form bots [8], disabling them in their original purpose and severely disrupting IoT industry operations. With vast amounts of Internet traffic data collected by security appliances, it is possible to reveal novel behavior of bots using data analysis algorithms. Many aspects of botnets have been researched quite thoroughly [1–3], such as detection of botnets and communication patterns

between bots and command/control (C&C) servers, but these are all technical studies, while few researches consider the perspective of the thriving botnet industry, which conducts cybersecurity attacks as a service.

The botnet industry seeks a high volume of DDoS attacks botnets can perform at any given time, low cost, and resistance to detection algorithms. These goals can be better achieved by organizing bots with unconventional methods, such as flexible organization.

Here are several scenarios that demonstrate the advantages of flexibly organizing bots from the point of view of the botnet industry: (1) Bots can be controlled as multiple small botnets with distinct technical properties – such as separate C&C servers and different C&C protocols – to evade detection. Many detection algorithms classify large groups of confirmed bots with identical technical properties as a botnet, so these small botnets with distinct properties are much harder to completely detect, and therefore have much better survivability. One way of implementing this is through the “super-botnet” structure proposed and analyzed by Vogt et al. [3]. (2) An attacker can utilize deceitful, advanced attack strategies, which are much more costly and time-consuming to defend against. Botnets could take distinct roles in a composite attack strategy. A known composite strategy is the usage of DDoS attacks as smoke-screens [4, 5] to draw defenders’ attention and cover up other attacks. (3) Bots in places where it is night may be turned off. Making bots in places where it is day attack together allows for maximum guaranteed attack volume.

We recognize that the term botnet is not enough to describe the organization of bots. The definition of botnets is from a technical perspective, yet these advantageous scenarios of organizing bots can be achieved in many technical ways: by creating a network of small botnets each with its own C&C server, by dividing a large botnet into separately managed portions, and more. Therefore, the concept “botnet” is ill-suited at describing these new methods.

This necessitates a new, broad, industry-oriented concept that describes these ways of organizing bots. We propose the concept of IP Gang to meet this demand.

Analyzing IP Gangs allows for smarter, strategic defences. Analysis from the perspective of the cyber-crime industry allows defenders to study, truly understand, and most importantly strategically defend against the behaviors of the attackers. For example, attackers have threatened to launch attacks unless a ransom is paid [7], and the defender can decide to pay or not in a more informed manner thanks to the additional knowledge on the IP Gang. In another situation, if some bots belonging to an IP Gang starts attacking, it would save precious time to immediately quarantine other bots of the IP Gang.

In this paper, we proved the existence of IP Gangs in real life Internet traffic, and developed a fast, high-compatibility detection algorithm that can be deployed in wide-scale, industrial applications to effectively defend against IP Gangs.

For compatibility, we only use the start time, source IP address, and target IP address describing events in easily-obtainable Internet event log data, which widely deployed network security appliances generally output. Other parameters describing the events (such as bytes per packet) are optional, and may help with accuracy if present.

The algorithm is based on the principle of single-linkage clustering [6], but with an additional “packaging” step that reduces the number of nodes to be clustered to increase speed. The detection algorithm outputs each detected IP Gang as a list of IP addresses. The complexity of the algorithm is $O(n^2)$ with a small constant, where n is the number of events in the data. Our test on one month’s events from a DDoS attack log detected 21 IP gangs, and showed that our algorithm is fast enough to be used to detect and help defend against IP Gangs on a large scale.

The rest of this paper is structured as follows. Section 2 presents previous work on botnet structure and botnet detection with Internet traffic. In Sect. 3, the formal definition of IP Gang is introduced and its relationship to botnets is analyzed. The principle and algorithm used to detect IP Gangs are detailed in Sect. 4. Next, the test results on real data are presented. We conclude by discussing future work.

2 Related Work

Botnet structures that may provide advantages similar to those of IP Gang’s have been studied. Most notably, Vogt et al. [3] proposed the “super-botnet”, a network of small, centralized botnets that can perform coordinated attacks, and provided detailed technical analysis of super-botnets. Individual botnets in a super-botnet can be detected, but it is very hard to detect the entire super-botnet. This additional resilience allows attackers to accumulate enough bots to perform very large-scale attacks. However, Vogt et al. did not provide experimentation on real life data. The concept of super-botnet is possibly related to the concept of IP Gang, but is still fundamentally different - it is still a technical definition, while the definition of IP Gang is business-oriented.

There had been a lot of researches on the detection of botnets based on Internet traffic. Gu et al. [1] was one of the first to propose and test on real life traffic data a clustering-based botnet detection model, which provides a variety of advantages over previous models that detect botnets by scanning for Command and Control (C&C) traffic between bots and the attacker. Gu et al. provided experimentation on real life data and analysis of the results.

3 Definition of IP Gang

As discussed in Sect. 1, the concept of IP Gangs and botnets are not comparable. IP Gangs and botnets consider the business and technical perspectives of groups of bots respectively. The former is concerned with how the attacker organizes his bots to his advantage, while the latter is instead mainly concerned with how the bots communicate with each other and with the controller.

Definition: *An IP Gang is a group of malware-infected computers (bots) that are controlled by the same attacker and often launch attacks directed at the same target within a short period of time t .*

A botnet is a group of bots that are organized by a certain network architecture and controlled by the same C&C (Command and Control) protocol by a logically centralized C&C server. Usually, the bots in a botnet have the same behavior from a technical point of view.

It is worth noting that, by definition, all the bots in a botnet always receive the same command, while bots in an IP Gang are not subject to such constraint.

The concept of IP Gang is suitable for describing the spatial and temporal features of organized bots, which could be in a same botnet or belongs to different botnets (Fig. 1). IP Gangs can be intentionally formed by attackers, or unintentionally formed due to logistic conditions. Bots of a botnet that are located in the same time zone may frequently be available to attack at the same time, thereby qualifying as an IP Gang.

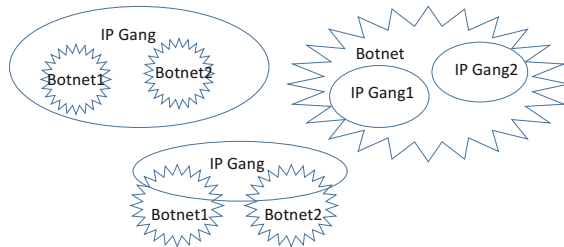


Fig. 1. IP gang and botnet

4 IP Gang Detection Algorithm

4.1 Overview

The target of the algorithm is to cluster the IP addresses of bots based on cybersecurity event log data to detect IP Gangs. We designed the method to meet the following challenges:

- (1) *Speed: the method must be fast enough to be deployed on networks which produce large volumes of log data.*
- (2) *Use as little information as possible: to ensure compatibility, the method must use only the most basic attack event information found in virtually all event logs: start time, source IP address, and destination IP address.*

At the core of the detecting algorithm is a clustering algorithm inspired by single-linkage clustering. Clustering algorithms are inherently time-intensive, and ours yields a complexity of $O(N^2)$, where N is the number of nodes to be clustered. Subsequently, reducing the number of nodes is crucial to the speed of the detection algorithm, and we add a packaging step before the clustering step to accomplish this. The algorithm consists of three steps, as illustrated in Fig. 2.

- (1) *Packaging: Events reported by security devices are grouped into Organized Attack Events (OAEs).*

- (2) *Clustering*: OAEs are clustered using a method inspired by single-linkage clustering with each finished cluster, named OAE cluster, representing a Gang.
- (3) *Analyzing*: OAE clusters are analyzed to find Gangs of IP addresses. This can also be seen as a “de-packaging” step, extracting IP addresses from OAE clusters.

Each step of our procedure is analyzed in greater detail in the rest of this section, and the performance of the procedure when run on real life data is discussed in the experimentation section.

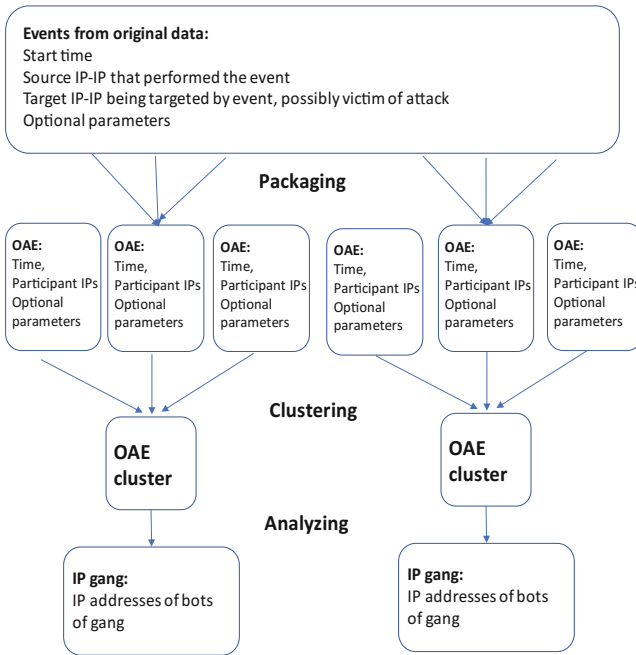


Fig. 2. Procedure of IP gang detection algorithm

4.2 Packaging: Constructing Organized Attack Events (OAEs) from Original Data

To decrease the number of nodes that need to be clustered in the clustering stage, we designed a way of packaging individual events with the same target IP address and starting in a time interval t into entities called Organized Attack Events (OAEs), which will be nodes in the clustering step.

The definition of an OAE is:

Given that A is a group of attack events, A_{IP} is the set of all of the source IP address in A . A is an OAE iff

$$\begin{aligned}
 &\forall a_1, a_2 \in A, \\
 &|a_1.starttime - a_2.starttime| \leq t \\
 &\& a_1.targetIP == a_2.targetIP \\
 &\& |A.IP| \geq min_size
 \end{aligned} \tag{1}$$

The set A_{IP} contains the IP addresses of all the bots that launched this set of Organized Attack Events (OAE).

The optimal value of time t varies between log data types.

Generally, attackers will simultaneously utilize a large number of bots in each organized attack for maximum effectiveness. Here we use this property to keep organized attacks – which are of use to us – and discard unorganized, individual attacks by filtering out OAEs with less than min_size events in them.

The pseudocode for the packaging process is:

```

1 id=0, and cur_OAE is initialized to empty
2 for event in D: #D denotes the input data
3   if(|event.start_time-cur_OAE.start_time|<=1 minute)...
4     and (event.target_IP==cur_OAE.target_IP):
5       cur_OAE.size+=1
6       cur_OAE.IP.append(event.source_IP)
7       optional_params_tmp.append(event.optional_params)
8   else:
9     if(cur_OAE.size>=minimum_size): #OAE is disposed if not big enough
10      cur_OAE.optional_params=take_most_common_values(optional_params_tmp)
11      #if the most common set of values for the
12      #optional params do not account for at least
13      #50% of the entries in the optional_params_tmp
14      #list, cur_OAE.optional_params is set to None
15
16      A.append(cur_OAE)
17      cur_OAE.id=id
18      id+=1
19      cur_OAE.size=1 #initialization with only current event
20      cur_OAE.IP=[event.source_IP]
21      cur_OAE.optional_params=None
22      cur_OAE.start_time=event.start_time
23      cur_OAE.target_IP=event.target_IP
    
```

At the start, the set of current OAE, cur_OAE , is initialized to contain only the first event. Afterwards, if the event being processed starts within t of cur_OAE , and has the same target IP address, it is added to cur_OAE . cur_OAE is added to the list of OAEs if the number of events it contains exceeds min_size . Otherwise, it is discarded and re-initialized.

This step has linear time complexity, and therefore is insignificant in terms of runtime. However, by using OAEs, instead of individual events in the clustering step, we decreased the number of nodes to be clustered by a very large factor, without sacrificing accuracy.

4.3 Clustering: Clustering OAEs to Form OAE Clusters

The OAEs formed in the packaging step are then clustered to form OAE clusters. Given A_1, A_2 are two OAEs, $A_{1,IP}$ as the set of all of the source IP address in A_1, A_1 and A_2 must be put in the same cluster if:

$$s = \frac{|A_{1,IP} \cap A_{2,IP}|}{|A_{1,IP}|} \geq \textit{combining_threshold}, \quad (2)$$

assuming $|A_{1,IP}| \leq |A_{2,IP}|$

In (2), s measures the normalized similarity of bots in two OAEs. Two OAEs with similarity larger than *combining.threshold* should be put in the same OAE cluster.

The clustering algorithm is inspired by single-linkage clustering, but is different in a crucial way. Single linkage clustering merges the two most similar clusters in each merge step, and stops performing merge steps when a reasonable total number of clusters have been reached. In contrast, we perform all possible merges of OAE clusters satisfying Eq. (2). In words, the clustering algorithm merges pairs of clusters that contain OAEs with a similarity score higher than the combining threshold but not yet in the same cluster. We proceed until no such pair exists.

The clustering algorithm is performed with a disjoint set data structure. For each OAE, the clustering algorithm computes the similarity scores between this OAE and all other OAEs. If the similarity score of two OAEs is higher than the combining threshold and the two OAEs are not yet in the same OAE cluster, the OAE clusters of the two OAEs are merged with a *union* operation.

The pseudo code is as follows:

```

1 for OAE A1 in set of all p OAEs:
2   count={} #counts the number of IP addresses
3   #each OAE has with the current OAE
4   for each IP address ip1 in A1.IP:
5     relevant=find(every OAE that contains ip1)
6     #The time this operation takes is near
7     #constant when a key-value database is used
8     #Each entry in "relevant" is an OAE that has
9     #at least 1 IP address in common with A1
10
11    #The number of entries in "relevant" is
12    #proportional to q, the average number
13    #of OAEs an IP address contributes to
14    for OAE A2 in relevant:
15      if(A1 and A2 are already in same cluster):
16        skip iteration
17      if(A2.id is in count):
18        count[A2.id]+=1
19      else:
20        count[A2.id]=1
21    for A2 in count:
22      if(A1 and A2 are already in same cluster):
23        continue
24      if(|A1.IP|<=|A2.IP|):
25        s=count[A2]/|A1.IP|
26      else:
27        s=count[A2]/|A2.IP|
28      if(s>=combining threshold):
29        merge the OAE clusters A1 and A2 are in
30          with "union" operation

```

In short, the clustering step puts OAEs that are performed by bots of the same gang into the same OAE cluster. This is achieved through computing the percentage of participating IP addresses two OAEs have in common and then merging the clusters the two OAEs are in if the percentage is higher than a threshold.

A NOSQL database is integral to our clustering method, as it greatly speeds up our method. With a relational database, the query at line 4 is very time-consuming, but NOSQL databases can perform this in a near constant time.

The complexity of this implementation is $O(n^2)$, where n is dataset size – the number of individual events in the log data. The number of iterations in the *for* loop in line 1 is p , the total amount of OAEs. The *for* loop in line 3 is independent from data size. The number of iterations in the *for* loops of lines 14 and 21 are both q , the average number of OAEs an IP address contributes to. Therefore, the overall complexity is $O(pq)$. It is apparent that $p \propto n$, as the average number of events in each OAE does not change. We expect $q \propto n$, though the correlation between the two is less strong. Therefore, $pq \propto n^2$, and the complexity is $O(n^2)$.

4.4 Analyzing: Identifying Gangs by Analyzing OAE Clusters

After OAE clusters are formed in the previous step, we analyze the OAE clusters to identify gangs of bots, with each bot represented by an IP address. We collect all the IP addresses that have participated in an OAE of an OAE cluster, and only retain IP addresses that have participated in enough OAEs of that cluster.

In words, for each OAE cluster, we calculate the percentage of OAEs in the cluster each IP address contributed to. IP addresses that only contribute to a very small percentage of OAEs may be treated as noise, as they are generally not worthy of studying. A threshold named *validation.threshold* is set in Sect. 5 of this paper and only the IP address with a contribution percentage larger than the threshold is retained into a gang.

5 Experimentation on Real Life Data

5.1 Overview

The data we used for experimentation is a DDoS log consisting of individual DDoS attack events collected from January 1st, 2016 to January 31st, 2016. The reports of DDoS attacks are collected from several dozens of NSFOCUS Network Traffic Analyzers (NTAs) and Anti-DDoS Systems (ADSS). NTAs and ADSs are deployed at the sites of the customers of NSFOCUS, and construct the DDoS log by analyzing netflow, an industry standard type of metadata.

Our algorithm was written in Python, used the Neo4j graph database, and was ran on a 2012 Thinkpad X230i laptop with hyper-threading disabled. The clustering step took 48 h with the full 1.5 GB of data, and the other steps were insignificant in terms of runtime.

With a *validation.threshold* of 0.05 and a *combining.threshold* of 0.6, our algorithm detected 17350 OAEs and 21 IP Gangs that has at least 10 OAEs. In total, there are 13916 valid bots in all these gangs.

On average, each OAE contained 183 individual events. This means that the packaging step decreased the runtime of the clustering step by a factor of 183^2 .

5.2 Discussion of the Parameters

In the “packaging” step, the optimal time t in Eq. (1) for our data is found to be 1 min. We observed in our log data that large groups of events that have the same target IP address typically have start times within 1 min of each other. Running the packaging step on our data with several different t values confirm $t = 1 \text{ min}$ as the optimal value. We determined the optimal value of *min_size* in Eq. (1) to be 50 with a similar procedure. In fact, we conducted tests with $t = 1, 3, 5 \text{ min}$ and *min_size* = 20, 30, 50, achieving very similar results in each test. We therefore chose $t = 1 \text{ min}$ and *min_size* = 50 to maximize accuracy.

In the “Analyzing” step, the value of *validation.threshold* greatly influences the final output of IP addresses in an IP gang. As shown in Fig. 3, different values of *validation.threshold* resulted in large variations in the total number of IP addresses in these 21 gangs. *Validation.threshold* provides a mechanism to look into an IP gang in different levels of granularity. For example, a larger *validation.threshold* will only output core members of an IP gang so that the defender could monitor the IP gang more efficiently. On the other hand, a smaller threshold will help the defender to get more detailed information on an IP gang.

5.3 Visualization and Discussion

We developed a type of diagram that visualizes the attack patterns of IP Gangs, which we denote the “attack fingerprint”. Each attack fingerprint represents a Gang, and each red dot on attack fingerprint represents an individual attack event by a bot of the Gang. The *ID* numbers of OAEs are assigned in time order, so the Y-axis is practically a relative measure of time. The X-axis is the *ID* of the bot, so each column of the figure represents the temporal behavior of a bot. The fingerprints in Fig. 4(a), (b), and (c) have *validation.threshold* = 0.05, while the one in Fig. 4(d) has *validation.threshold* = 0.1.

In 6 of the 21 fingerprints, we see that all the columns have nearly the exact same appearance. This tells us that each bot in the gang participated in almost the same OAEs. The attack fingerprints in Fig. 4(c) is an example. This kind of fingerprints can be explained with the conventional concept of botnets, because all the bots are behaving in the same way.

The other 15 fingerprints are difficult to describe with only the concept of botnets, because the behavior of bots often differ from each other. As shown in Fig. 4(a) and (b), the columns take a small number of distinct but still similar appearances. In certain rows, all columns have red dots, but the columns take several distinct patterns in other rows. This shows that the bots are not always behaving in the same way. Notably, different values of *validation.threshold* allows for different parts of the Gang to be analyzed in detail. Figure 4(d) demonstrates this, as it describes the same Gang as

Effect of validation.threshold on the number of bots in Gangs

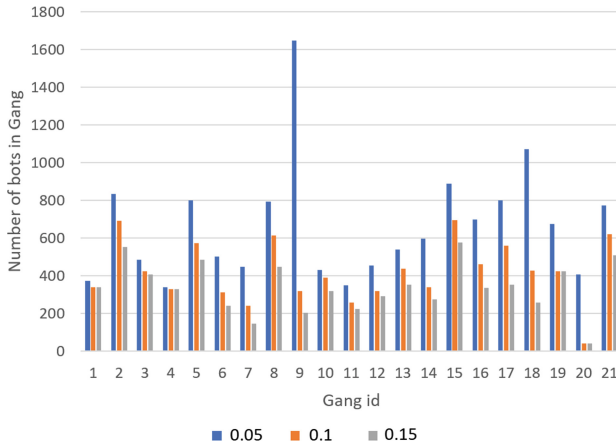


Fig. 3. Influence of validation.threshold

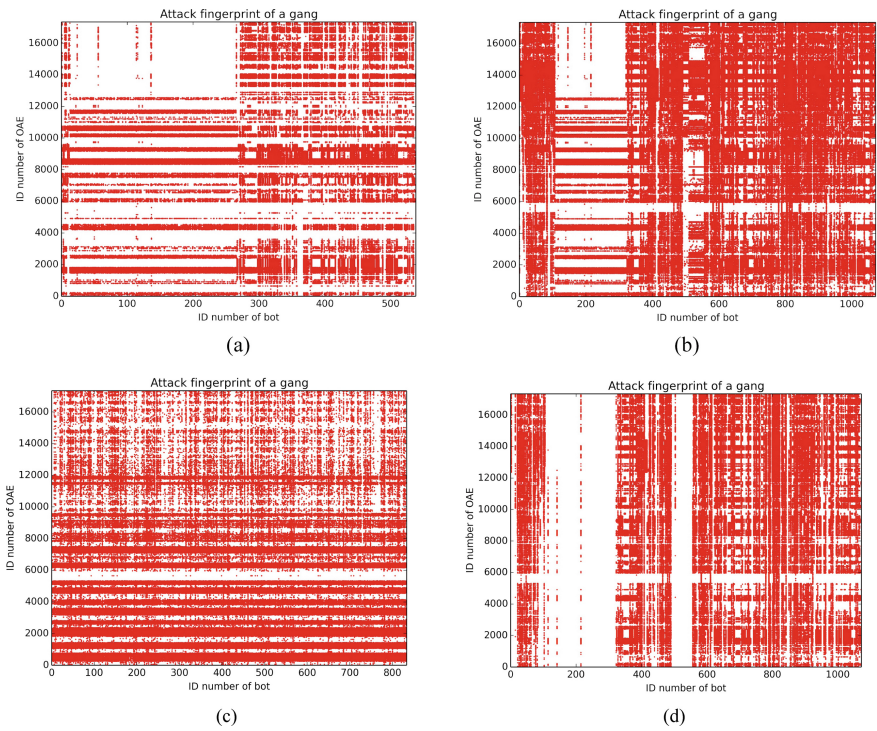


Fig. 4. Fingerprints of gangs (Color figure online)

Fig. 4(b), but is drawn with *validation.threshold* = 0.1. Clearly, the bots with *id* between 100 and 300 are omitted from the graph, while the other parts are preserved. There are several plausible explanations for this phenomenon. For example, in Fig. 3 (a), there may be two botnets, one with bots *ID* < 280, and another with bot *ID* > 280. Sometimes they attack together as shown by the dense horizontal lines, and sometimes they attack separately as shown by the upper part of the fingerprint with OAE *ID* > 12000. Another explanation is that these bots belong to the same botnet, but sometimes only part of the botnet are able to successfully carry out the attack.

6 Conclusions and Future Work

We demonstrate that IP gangs exist on the internet and are actively being used by attacker to perform DDoS attacks. They are detectable using clustering-based algorithms and can be distinguished from conventional botnets.

Analyzing the behavior of IP gangs will be highly beneficial. Doing so can make for a better understanding of the operation, structure, and performance of IP gangs. A more thorough understanding of these is necessary to accurately assess the threat that IP gangs pose to cybersecurity, and to defend against IP gangs. For defenders, knowing more about the behavior of Gangs can allow for smarter, strategic defences against Internet attacks.

References

1. Gu, G., Perdisci, R., Zhang, J., Lee, W.: BotMiner: clustering analysis of network traffic for protocol-and structure-independent botnet detection. In: USENIX Security Symposium, vol. 5, no. 2, pp. 139–154 (2008)
2. Khattak, S., Ramay, N.R., Khan, K.R., Syed, A.A., Khayam, S.A.: A taxonomy of botnet behavior, detection, and defense. *IEEE Commun. Surv. Tutor.* **16**(2), 898–924 (2014)
3. Vogt, R., Aycok, J., Jacobson, M.: Army of botnets. In: Proceedings of NDSS 2007 (2007)
4. Arbor Networks: DDoS as a smokescreen for fraud and theft, 3 February 2016. <https://www.arbornetworks.com/blog/insight/ddos-as-a-smokescreen-for-fraud-and-theft/>
5. Kaspersky Lab: Research reveals hacker tactics: Cybercriminals use DDoS as smokescreen for other attacks on business, 22 November 2016. https://www.kaspersky.com/about/press-releases/2016_research-reveals-hacker-tactics-cybercriminals-use-ddos-as-smokescreen-for-other-attacks-on-business
6. Stanford Natural Language Processing Group: Single-link and complete-link clustering (n.d.). <https://nlp.stanford.edu/IR-book/html/htmledition/single-link-and-complete-link-clustering-1.html>
7. WeLiveSecurity: Spammed-out emails threaten websites with DDoS attack on September 30th, 25 September 2017. <https://www.welivesecurity.com/2017/09/25/email-ddos-threat/>
8. Koliass, C., Kambourakis, G., Stavrou, A., Voas, J.: DDoS in the IoT: mirai and other botnets. *Computer* **50**(7), 80–84 (2017)
9. Pultarova, T.: Cyber security - Ukraine grid hack is wake-up call for network operators. *Eng. Technol.* **11**(1), 12–13 (2016)

10. Khan, R., Maynard, P., McLaughlin, K., Lavery, D., Sezer, S.: Threat analysis of BlackEnergy malware for synchrophasor based real-time control and monitoring in smart grid. In: Janicke, H., Jones, K., Brandstetter, T. (eds.) 4th International Symposium for ICS & SCADA Cyber Security Research 2016, pp. 53–63 (2016)
11. Kaspersky Lab: Attack on Dyn explained. <https://www.kaspersky.com/blog/attack-on-dyn-explained/13325/>



Medical AI System to Assist Rehabilitation Therapy

Takashi Isobe^{1,2,3(✉)} and Yoshihiro Okada²

¹ Hitachi High Technologies America, Inc., Pleasanton, CA 94588, USA
takashi.isobe.fm@hitachi.com

² Hitachi High-Technologies Solutions, Chuo-ku, Tokyo 104-6031, Japan

³ Hitachi, Ltd., Kokubunji, Tokyo 185-8601, Japan

Abstract. Value-based program and payment are becoming general in healthcare. In rehabilitation medicine, the medical services are becoming to be paid depending on the outcome of recovery called as FIM (Functional Independent Measurement) gain. Optimal combination of therapies classified to physical, occupational and speaking ones differs by each patient's age, sex, disease, handicap, time-series FIM score and therapies. Non-experienced hospitals and doctors have a difficulty in improving the outcome of recovery. Therefore, the demand for medical AI system to assist rehabilitation therapy is increasing. We developed a medical AI system to assist rehabilitation therapy. Our proposed system piled up an actual time-series medical record into matrixes or images, and recognized the pattern of patients' outcome using machine learning. The interface displayed not only the statistical information about similar patients but also the optimal combination of therapies with estimated FIM gain and probability by each patient. Our AI predicted the pattern of outcome from three patterns at the accuracy of 57.8% and at the response of 5–8 s using GPU. The pattern achieving high FIM gain, which was most important because it was used to suggest optimal combination of therapies, occupied the proportion of 31.6% in the actual medical record while the precision was 63.7% and the recall was 72.9%. Our AI system could correctly extract 72.9% of the most important pattern used as candidates for the suggestion.

Keywords: Medicine · Artificial Intelligence (AI) · Machine learning

1 Background and Purpose

Value-based program [1] and payment [2] are becoming general in healthcare. In rehabilitation medicine, the medical services are becoming to be paid based on the outcome of recovery determined by the combination of hospital days and FIM (Functional Independent Measurement) gain [3]. The outcome of recovery has an impact on the revenue of rehabilitation hospitals.

The rehabilitation therapies are composed of physical, occupational and speaking therapies (PT, OT and ST). Their optimal combination varies depending on many parameters such as patients' age, sex, disease, handicap, time-series FIM score and therapies. Non-experienced hospitals or doctors don't know the optimal combination

well and has a difficulty in improving the performance of recovery. Therefore, the demand for medical AI system to assist rehabilitation therapy is increasing. We aimed at developing a medical AI system to assist rehabilitation therapy.

2 Traditional AI

Medical diagnostic using correlation [4] is the most popular technique. If the distribution of data is small and correlation coefficient is large, correlation base diagnostic generally has high accuracy. In rehabilitation medicine, the distribution of FIM gain against each parameter is large and the correlation coefficient often becomes small between FIM gain and many parameters because humans don't react quantitatively like machines.

Medical diagnostic using bayesian network [5] is also well known. If each conditional branch node has the table of conditional probability defined by optimal condition, diagnostic using bayesian network has high accuracy. Rehabilitation's outcome differs widely by the difference of a few percentage in the allocation of therapies between PT, OT and ST. It is often difficult to find the optimal condition in rehabilitation medicine.

3 Proposed AI System to Assist Therapy for Rehabilitation

3.1 System Configuration

The following Fig. 1 shows the configuration of our proposed AI system to assist therapy for rehabilitation. Customers send the electric medical data to AI application of cloud, and after then, they receive the result of optimal therapies and predicted outcomes. The electric medical data includes patient's information with latest therapies and FIM scores. The predicted outcomes include FIM gain and hospital days.

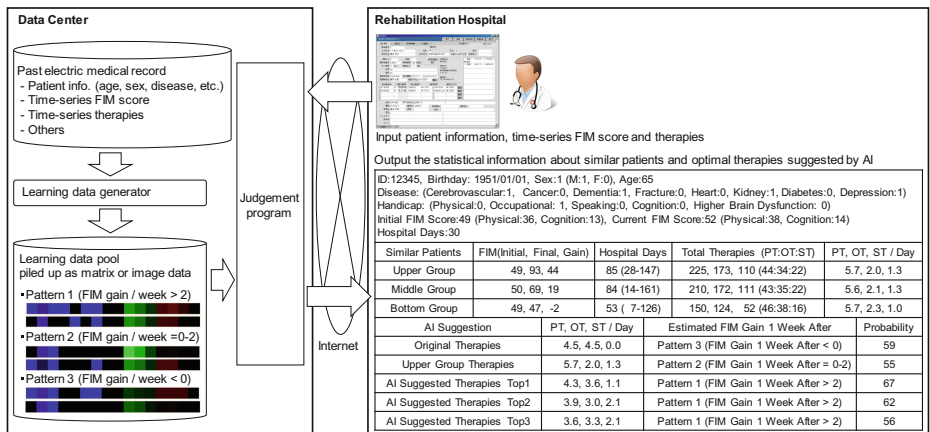


Fig. 1. System configuration

AI application of cloud is composed of two programs. One is learning data generator. Another is judgement program. Learning data generator creates the learning data piled up as vector and image type data using the archives of the past electric medical record composed of patients' personality, time-series FIM scores and therapies. It classifies the data into multiple patterns depending on FIM gain. Judgement program recognizes the pattern of patients' recovery by machine learning using learning data. It outputs the statistical information about similar patients and suggests the optimal therapies combined with the estimated FIM gain and the probability.

3.2 Medical Record Data Used for Learning

Through the collaboration with an actual rehabilitation hospital [6], we analyzed the actual electric medical record including eighteen thousand patients between 2002 and 2017. Our system classified diseases into eight categories (cerebrovascular, heart, kidney, diabetes, cancer, dementia, bone fracture, depression) and handicaps into five categories (physical, speaking, occupation, cognition, higher brain dysfunction).

The statistical analysis of the actual record shows the trend where highly recovered patient group receives larger number of PT/OT and smaller number of ST as initial FIM score or age becomes larger. On the contrary, too large number of PT/OT or too small number of ST causes the worse outcome. The type of disease also influences the outcome. Cerebrovascular shows the trend where highly recovered patient group receives larger number of ST. On the other hand, heart disease or cancer shows the trend where highly recovered patient group receives larger number of PT/OT.

3.3 Learning Data Generation

Learning data generator creates the learning data piled up as vector and image type data composed of sex, age, disease, handicap, FIM score (physical, cognition and acceleration) and therapies (the number of PT, OT and ST) by each combination of patient's ID and hospitalized day. Vector type data is used for vector distance base algorithm. Image type data is used for deep learning algorithm using convolutional neural network.

FIM score and therapies are smoothed by each day between two days when FIM scores are measured. One row becomes one vector or one image data. The image data piles up sex, age, disease and handicap as blue pixels, FIM score as green pixels and therapies as red pixels (See Fig. 2).

Each vector or image data is classified into three patterns depending on FIM gain per one week. FIM gain of more than two per one week is defined as pattern 1. FIM gain of two or less and more than zero per one week is defined as pattern 2. FIM gain of zero or less per one week is defined as pattern 3. Pattern 1 occupies 31.6% of all data and 30.8% of data four days after hospitalization (See Table 1).

Classified to pattern 1: if FIM gain per week is more than 2
 pattern 2: if FIM gain per week is between 0 and 2
 pattern 3: if FIM gain per week is 0 or less

Patient ID	Hospitalized Day	FIM Measurement Day	Sex	Age	Disease			Handicap			FIM			Therapies / Day			Pattern		
					Cerebrovascular	Heart	Kidney	Physical	Occupational	Speaking	Physical	Cognition	Acceleration	PT	OT	ST			
12345	12/20/2016	12/22/2016	1	65	1	0	1	...	0	1	0	...	36	13	-	4	4	1	1
12345	12/20/2016		1	65	1	0	1	...	0	1	0	...	37	13.5	-	4	4	1	1
12345	12/20/2016	12/24/2016	1	65	1	0	1	...	0	1	0	...	38	14	1.5	3	5	1	2
12345	12/20/2016		1	65	1	0	1	...	0	1	0	...	38.25	14	1.5	3	5	1	2
12345	12/20/2016		1	65	1	0	1	...	0	1	0	...	38.50	14	1.5	3	5	1	2
12345	12/20/2016		1	65	1	0	1	...	0	1	0	...	38.75	14	1.5	3	5	1	2
12345	12/20/2016	12/28/2016	1	65	1	0	1	...	0	1	0	...	39	14	0.25	4	5	0	3
12345	12/20/2016		1	65	1	0	1	...	0	1	0	...	39	13.5	0.25	4	5	0	3
12345	12/20/2016	12/30/2016	1	65	1	0	1	...	0	1	0	...	39	13	-0.5	-	-	-	-

Smooth the change of FIM score and the accumulated number of therapies between two FIM measurement days

Fig. 2. Learning data

Table 1. Proportion of each pattern in data learned from actual medical record

Data	Pattern 1	Pattern 2	Pattern 3
All	239788 (31.6%)	325181 (42.8%)	194492 (25.6%)
Four days after hospitalization	224256 (30.8%)	315896 (43.3%)	188973 (25.9%)

3.4 Pattern Recognition

Judgement program extracts top 700 of most similar vectors from all learned vectors using Euclidean distance composed of sex, age, disease, handicap, FIM score and therapies between original patient’s vector and all learned vectors. The program calculates the proportion of pattern 1–3 in the extracted top 700 and outputs the pattern deviating significantly from the proportion of learning data shown in Table 1 as the recognized pattern.

In the evaluation, we also used deep learning technique using image data instead of vector data for comparison to vector distance technique. The deep learning technique used convolutional neural network with two layers enough to maximize accuracy.

3.5 Suggestion of Optimal Therapies

Judgement program extracts top 300 of most similar vectors from vectors of pattern 1 using Euclidean distance composed of sex, age, disease, handicap and FIM score between original patient’s vector and all learned vectors. The program newly creates 300 vectors by merging the therapies of the top 300 similar vectors with sex, age, disease, handicap and FIM score of original patient. The patterns of newly created 300 vectors are recognized 300 times. The vectors recognized as pattern 1 are sorted in descending order of probability, and the therapies of the top 3 vectors are output as

suggested therapies with the probability. If there is no vector recognized as pattern 1, the vectors recognized as pattern 2 or 3 are used. The vectors recognized as pattern 3 are sorted in ascending order of probability.

4 Evaluation

We divided the actual electric medical record into two groups of 99.8% for learning and 0.2% for evaluation, and then evaluated the accuracy of pattern recognized from three patterns using two techniques: vector distance and deep learning [7].

Vector distance achieved the accuracy of 57.8% that was larger than that of 49.6% achieved by deep learning (See Table 2). Vector distance uses exact boundary conditions based on brute-force distances. On the other hand, deep learning uses approximated boundary conditions. This difference was thought to make the accuracy of vector distance higher than that of deep learning.

Vector distance technique can also estimate the time-series FIM gain and hospital days by averaging the extracted similar vectors and show them as additional information. It has high extensibility and our system used vector distance technique.

Table 2. Accuracy of pattern recognition

Pattern recognition technique	Accuracy
Vector distance	57.8%
Deep learning	49.6%

Pattern 1 achieving high FIM gain, which was most important because it was used to predict optimal combination of therapies, occupied the proportion of 31.6% in the actual record while the precision was 63.7% and the recall was 72.9% (See Table 3). Our AI system could correctly extract 72.9% of the most important pattern 1 used as candidates for the suggestion. Achieving even higher accuracy close to 100% is challenging because patients have emotions and react non-quantitatively unlike machines.

Table 3. Prediction of most important pattern 1

Pattern 1	Evaluated value
Proportion in actual medical record	31.6%
Precision	63.7%
Recall	72.9%

Improvement of execution time using GPU [8] is shown in Table 4. The execution time increased in proportion to the number of similar vectors extracted for suggestion. The top 300 or less satisfied the response of 10 s required by our customer.

Table 4. Comparison of execution time between CPU [9] and GPU

Similar vectors extracted for suggestion	CPU (second)	GPU (second)
Top 100	15	5
Top 300	29	8
Top 700	68	17

5 Conclusion

Our rehabilitation medical AI system created learning data by piling up the past electric medical record as the matrix or image data, and recognized the pattern of outcome using machine learning. The interface displayed the optimal combination of therapies with estimated FIM gain and probability. Our AI predicted the pattern of outcome from three patterns at the accuracy of 57.8% and at the response of 5–8 s using GPU. Pattern 1 achieving high FIM gain, which was most important because it was used to predict optimal combination of therapies, occupies the proportion of 31.6% in the actual medical record while the precision was 63.7% and the recall was 72.9%. Our AI system could correctly extract 72.9% of the most important pattern 1 used as candidates for the suggestion. For the future commercialization, we will keep proof-of-concept on going in an actual hospital and improve our system.

References

- Centers for Medicare & Medicaid Services' (CMS') Value-Based Programs. <https://www.cms.gov/Medicare/Quality-Initiatives-Patient-Assessment-Instruments/Value-Based-Programs/Value-Based-Programs.html>
- HCI's Value-Based Payment. <http://www.hci3.org/tools-resources/metrics-tracking-transformation-us-health-care/value-based-payment-metrics-transformation>
- Ministry of Health, Labour and Welfare's Homepage. <http://www.mhlw.go.jp/file/05-Shingikai-12404000-Hokenkyoku-Iryouka/0000169318.pdf>
- Mukaka, M.M.: A guide to appropriate use of correlation coefficient in medical research. *US Natl. Libr. Med.* **24**(3), 69–71 (2012)
- Nikovski, D.: Constructing Bayesian networks for medical diagnosis from incomplete and partially correct statistics. *IEEE Trans. Knowl. Data Eng.* **12**(4), 509–516 (2000)
- Hatsudai Rehabilitation Hospital. <http://www.hatsudai-reha.or.jp/>
- Google TensorFlow Homepage. <https://www.tensorflow.org/>
- Nvidia GPU P100 Homepage. <http://www.nvidia.com/object/tesla-p100.html>
- Intel CPU E5-1620 v3 Homepage. https://ark.intel.com/products/82763/Intel-Xeon-Processor-E5-1620-v3-10M-Cache-3_50-GHz



A Novel Parallel Algorithm for Frequent Itemsets Mining in Large Transactional Databases

Huan Phan^{1,2(✉)} and Bac Le³

¹ Division of IT, University of Social Sciences and Humanities,
VNU-HCM, Ho Chi Minh City, Vietnam
huanphan@hcmussh.edu.vn

² Faculty of Mathematics and Computer Science, University of Science,
VNU-HCM, Ho Chi Minh City, Vietnam

³ Faculty of IT, University of Science, VNU-HCM, Ho Chi Minh City, Vietnam
lhbac@fithcmus.edu.vn

Abstract. Since the era of data explosion, data mining in large transactional databases has become more and more important. There are many data mining techniques like association rule mining, the most important and well-researched one. Furthermore, frequent itemset mining is one of the fundamental but time-consuming steps in association rule mining. Most of the algorithms used in literature find frequent itemsets on search space items having at least a *minsup* and are not reused for subsequent mining. Therefore, in order to decrease the execution time, some parallel algorithms have been proposed for mining frequent itemsets. Nonetheless, these algorithms merely implement the parallelization of Apriori and FP-Growth algorithms. To deal with this problem, several parallel NPA-FI algorithms are proposed as a new approach in order to quickly detect frequent itemsets from large transactional databases using an array of co-occurrences and occurrences of kernel item in at least one transaction. Parallel NPA-FI algorithms are easily used in many distributed file system, namely Hadoop and Spark. Finally, the experimental results show that the proposed algorithms perform better than other existing algorithms.

Keywords: Association rules · Co-occurrence items · Frequent itemsets
Parallel algorithm

1 Introduction

Mining frequent itemsets is a fundamental and essential problem in many data mining applications such as the discovery of association rules, strong rules, correlations, multi-dimensional patterns, and many other important discovery tasks. The problem is formulated as follows: Given a large database of set of items transactions, find all frequent itemsets, where a frequent itemset is one that occurs in at least a user-specified percentage of transaction database [4].

In the last three decades, most of the mining algorithms for frequent itemsets, proposed by various authors around the world, are based on Apriori [5] and FP-Tree [6, 9]. Simultaneously to speed up the implementation of the mining frequent itemsets, authors worldwide propose the parallelization of algorithms based on the Apriori [1, 7] and FP-Tree [8]. In this paper, we propose a novel sequential algorithm that mines frequent itemsets, and then, parallelizing the sequential algorithm to demonstrate the multi-core processors in an effective way as follows.

- **Algorithm 1:** Computing Kernel_COOC array of co-occurrences and occurrences of kernel item in at least one transaction;
- **Algorithm 2:** Generating all frequent itemsets based on Kernel_COOC array;
- Parallel **NPA-FI** algorithm quickly mining frequent itemsets from large transactional databases implemented on the multi-core processors.

This paper is organized as follows: in Sect. 2, we describe the basic concepts for mining frequent itemsets, benchmark datasets description and data structure for transaction databases. Some theoretical aspects of our approach relies, are given in Sect. 3. Besides, we describe our sequential algorithm to compute frequent itemsets on large transactional databases. After that we parallelize the proposed sequential algorithm. Details on implementation and experimental tests are discussed in Sect. 4. Finally, we conclude with a summary of our approach, perspectives and extensions of this future work.

2 Background

2.1 Frequent Itemset Mining

Let $I = \{i_1, i_2, \dots, i_m\}$ be a set of m distinct items. A set of items $X = \{i_1, i_2, \dots, i_k\}, \forall i_j \in I (1 \leq j \leq k)$ is called an itemset, an itemset with k items is called a k -itemset. \mathbf{D} be a dataset containing n transaction, a set of transaction $T = \{t_1, t_2, \dots, t_n\}$, and each transaction $t_j = \{i_{k_1}, i_{k_2}, \dots, i_{k_l}\}, \forall i_{k_l} \in I (1 \leq k_l \leq m)$.

Definition 1. The support of an itemset X is the number of transaction in which occurs as a subset, denoted as $sup(X)$.

Definition 2. Let $minsup$ be the threshold minimum support value specified by user. If $sup(X) \geq minsup$, itemset X is called a frequent itemset, denoted FI is the set of all the frequent itemset.

Property 1. $\forall X \subseteq Y$, If $sup(Y) \geq minsup$ then $sup(X) \geq minsup$.

Property 2. $\forall X \subset Y$, If $sup(X) < minsup$ then $sup(Y) < minsup$.

See an example transaction database \mathcal{D} in Table 1.

Table 1. The transaction database \mathcal{D} used as our running example.

TID	Items							
t1	A	C	E	F				
t2	A	C			G			
t3			E		H			
t4	A	C	D	F	G			
t5	A	C	E	G				
t6						E		
t7	A	B	C	E				
t8	A	C	D					
t9	A	B	C	E	G			
t10	A	C	E	F	G			

Table 2. Mining frequent itemsets.

k-itemset	FI (minsup = 2)	FI (minsup = 3)	FI (minsup = 5)
1	D, B, F, G, E, A, C	F, G, E, A, C	G, E, A, C
2	BE, BA, BC, DA, DC, FE, FG, FA, FC, GE, GA, GC, EA, EC, AC	FA, FC, GE, GA, GC, EA, EC, AC	GA, GC, EA, EC, AC
3	BAC, BEA, DAC, FEA, BEC, FEC, FGA, CFG, FAC, GEA, GEC, EAC, GAC	FAC, GEA, GEC, GAC, EAC	GAC, EAC
4	BEAC, FGAC, FEAC, GEAC	GEAC	

Example 1. See Table 1. There are eight different items $I = \{A, B, C, D, E, F, G, H\}$ and ten transactions $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}\}$. Table 2 shows the frequent itemsets at three different minsup values—2 (20%), 3 (30%) and 5 (50%) respectively.

2.2 Benchmark Description

Djenouri *et al.* categorized the datasets: Three types of well-known instance details the characteristic of these benchmarks [10].

Table 3. Datasets description.

Instance type	#Trans	#Items	#Avg. length
Medium	6,000 to 9,000	500 to 16,000	2 to 500
Large	100,000 to 500,000	1,000 to 1,600	2 to 10
Big	up 1,600,000	up 500,000	

2.3 Data Structure for Transaction Database

The binary matrix is an efficient data structure for mining frequent itemsets [2, 3]. The process begins with the transaction database transformed into a binary matrix BiM, in

which each row corresponds to a transaction and each column corresponds to an item. Each element in the binary matrix BiM contains 1 if the item is presented in the current transaction; otherwise it contains 0 (Fig. 1).

TID	A	B	C	D	E	F	G	H
t1	1	0	1	0	1	1	0	0
t2	1	0	1	0	0	0	1	0
t3	0	0	0	0	1	0	0	1
t4	1	0	1	1	0	1	1	0
t5	1	0	1	0	1	0	1	0
t6	0	0	0	0	1	0	0	0
t7	1	1	1	0	1	0	0	0
t8	1	0	1	1	0	0	0	0
t9	1	1	1	0	1	0	1	0
t10	1	0	1	0	1	1	1	0

Fig. 1. A binary matrix BiM representation of example transaction database.

3 The Proposed Algorithms

3.1 Generating Array of Co-occurrence Items of Kernel Item

In this part, we illustrate the framework of the algorithm generating co-occurrence items of items in transaction database.

Definition 3. Project set of item i_k on database $\mathcal{D} : \pi(i_k) = \{t_j \in \mathcal{D} | i_k \subseteq t_j\}$ is set of transaction contain item i_k (π -decreasing monotonic). According to Definition 1:

$$sup(i_k) = |\pi(i_k)| \tag{1}$$

Example 2. See Table 1. Consider item B , we detect project set of item B on database $\mathcal{D} : \pi(B) = \{t_7, t_9\}$ then $sup(B) = |\pi(B)| = 2$.

Definition 4. Project set of itemset $X = \{i_1, i_2, \dots, i_k\}, \forall i_j \in I : \pi(X) = \pi(i_1) \cap \pi(i_2) \dots \pi(i_k)$.

$$sup(X) = |\pi(X)| \tag{2}$$

Example 3. See Table 1. Consider item E , we detect project set of item E on database $\mathcal{D} : \pi(E) = \{t_1, t_3, t_5, t_6, t_7, t_9, t_{10}\}$ then $sup(BE) = |\pi(BE)| = |\pi(B) \cap \pi(E)| = |\{t_7, t_9\} \cap \{t_1, t_3, t_5, t_6, t_7, t_9, t_{10}\}| = |\{t_7, t_9\}| = 2$.

Definition 5. Let $i_k \in I$ is called a kernel item. Itemset $X_{cooc} \subseteq I$ is called co-occurrence items with kernel item i_k , so that satisfy $\pi(i_k) \equiv \pi(i_k \cup X_{cooc})$. Denoted as $cooc(i_k) = X_{cooc}$.

Example 4. See Table 1. Consider item B as kernel item, we detect co-occurrence items with item B as $cooc(B) = \{A, C, E\}$ and $sup(B) = sup(BACE) = 2$.

Definition 6. Let $i_k \in I$ is called a kernel item. Itemset $Y_{looc} \subseteq I$ is called occurrence items with kernel item i_k in at least one transaction, but not co-occurrence items, so that satisfy $1 \leq |\pi(i_k \cup i_{looc})| < |\pi(i_k)|, \forall i_{looc} \in Y_{looc}$. Denoted as $looc(i_k) = Y_{looc}$.

Example 5. See Table 1. Consider item B as kernel item, we detect occurrence items with item B in at least one transaction $looc(B) = \{G\}$ and $\pi(BG) = \{t_9\} \subset \pi(B) = \{t_7, t_9\}$.

Algorithm Generating Array of Co-occurrence Items

This algorithm is generating co-occurrence items of items in transaction database and archive into the *Kernel_COOC* array. Each element within the *Kernel_COOC*, 4 fields:

- *Kernel_COOC*[k].item: kernel item k ;
- *Kernel_COOC*[k].sup: support of kernel item k ;
- *Kernel_COOC*[k].cooc: co-occurrence items with kernel item k ;
- *Kernel_COOC*[k].looc: occurrence items kernel item k in least one transaction.

The framework of Algorithm 1 is as follows:

Algorithm 1. Generating Array of Co-occurrence Items

Input : Dataset \mathcal{D}
Output: *Kernel_COOC* array, matrix *BiM*

- 1: **foreach** *Kernel_COOC*[k] **do**
- 2: *Kernel_COOC*[k].item = i_k
- 3: *Kernel_COOC*[k].sup = 0
- 4: *Kernel_COOC*[k].cooc = $2^m - 1$
- 5: *Kernel_COOC*[k].looc = 0
- 6: **foreach** $t_j \in T$ **do**
- 7: **foreach** $i_k \in t_j$ **do**
- 8: *Kernel_COOC*[k].sup ++
- 9: *Kernel_COOC*[k].cooc = *Kernel_COOC*[k].cooc **AND** vectorbit(t_j)
- 10: *Kernel_COOC*[k].looc = *Kernel_COOC*[k].looc **OR** vectorbit(t_j)
- 11: sort *Kernel_COOC* array in ascending by support

We illustrate Algorithm 1 on example database in Table 1.

Initialization of the *Kernel_COOC* array, number items in database $m = 8$;

Item	A	B	C	D	E	F	G	H
sup	0	0	0	0	0	0	0	0
cooc	11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111
looc	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

Read once of each transaction from $t1$ to $t10$
 Transaction $t1 = \{A, C, E, F\}$ has vector bit representation **10101100**;

Item	A	B	C	D	E	F	G	H
sup	1	0	1	0	1	1	0	0
cooc	10101100	11111111	10101100	11111111	10101100	10101100	11111111	11111111
looc	10101100	00000000	10101100	00000000	10101100	10101100	00000000	00000000

Transaction $t2 = \{A, C, G\}$ has vector bit representation **10100010**;

Item	A	B	C	D	E	F	G	H
sup	2	0	2	0	1	1	1	0
cooc	10100000	11111111	10100000	11111111	10101100	10101100	10100010	11111111
looc	10101110	00000000	10101110	00000000	10101100	10101100	10100010	00000000

Transaction $t3 = \{E, H\}$ has vector bit representation **00001001**;

Item	A	B	C	D	E	F	G	H
sup	2	0	2	0	2	1	1	1
cooc	10100000	11111111	10100000	11111111	00001000	10101100	10100010	00001001
looc	10101110	00000000	10101110	00000000	10101101	10101100	10100010	00001001

Transaction $t4 = \{A, C, D, F, G\}$ has vector bit representation **10110110**;

Item	A	B	C	D	E	F	G	H
sup	3	0	3	1	2	2	2	1
cooc	10100000	11111111	10100000	10110110	00001000	10100100	10100010	00001001
looc	10111110	00000000	10111110	10110110	10101101	10111110	10110110	00001001

Transaction $t5 = \{A, C, E, G\}$ has vector bit representation **10101010**;

Item	A	B	C	D	E	F	G	H
sup	4	0	4	1	3	2	3	1
cooc	10100000	11111111	10100000	10110110	00001000	10100100	10100010	00001001
looc	10111110	00000000	10111110	10110110	10101111	10111110	10111110	00001001

Transaction $t6 = \{E\}$ has vector bit representation **00001000**;

Item	A	B	C	D	E	F	G	H
sup	4	0	4	1	4	2	3	1
cooc	10100000	11111111	10100000	10110110	00001000	10100100	10100010	00001001
looc	10111110	00000000	10111110	10110110	10101111	10111110	10111110	00001001

Transaction $t7 = \{A, B, C, E\}$ has vector bit representation **11101000**;

Item	A	B	C	D	E	F	G	H
sup	5	1	5	1	5	2	3	1
cooc	10100000	11101000	10100000	10110110	00001000	10100100	10100010	00001001
looc	11111110	11101000	11111110	10110110	11101111	10111110	10111110	00001001

Transaction $t8 = \{A, C, D\}$ has vector bit representation **10110000**;

Item	A	B	C	D	E	F	G	H
sup	6	1	6	2	5	2	3	1
cooc	10100000	11101000	10100000	10110000	00001000	10100100	10100010	00001001
looc	11111110	11101000	11111110	10110110	11101111	10111110	10111110	00001001

Transaction $t9 = \{A, B, C, E, G\}$ has vector bit representation **11101010**;

Item	A	B	C	D	E	F	G	H
sup	7	2	7	2	6	2	4	1
cooc	10100000	11101000	10100000	10110000	00001000	10100100	10100010	00001001
looc	11111110	11101010	11111110	10110110	11101111	10111110	11111110	00001001

The last, transaction $t10 = \{A, C, E, F, G\}$ has vector bit representation **10101110**;

Item	A	B	C	D	E	F	G	H
sup	8	2	8	2	7	3	5	1
cooc	10100000	11101000	10100000	10110000	00001000	10100100	10100010	00001001
looc	11111110	11101010	11111110	10110110	11101111	10111110	11111110	00001001

After the processing of Algorithm 1, the Kernel_COOC array as follows:

Table 4. Kernel_COOC array are ordered in support ascending order.

Item	H	B	D	F	G	E	A	C
sup	1	2	2	3	5	7	8	8
cooc	E	A, C, E	A, C	A, C	A, C		C	A
looc		G	F, G	D, E, G	B, D, E, F	A, B, C, F, G, H	B, D, E, F, G	B, D, E, F, G

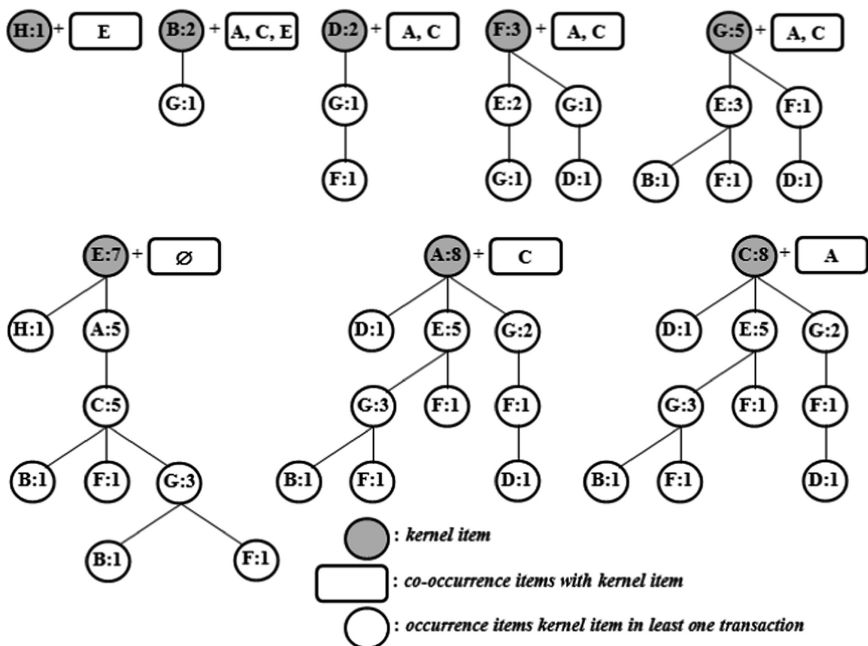


Fig. 2. The pattern-tree of occurrence items with kernel item in as least one transaction.

See Table 4 and Fig. 2, we have $cooc(A) = \{C\}$ and $cooc(C) = \{A\}$. In this case, the frequent itemset generated from A and C items will be duplicated. We provide a Definitions 7, 8 to eliminate the duplication when generating frequent itemsets.

Definition 7. Let $i_k \in I(i_1 \prec i_2 \prec \dots \prec i_m)$ items are ordered in support ascending order, i_k is called a kernel item. Itemset $X_{lexcooc} \subseteq I$ is called co-occurrence items with the kernel item i_k , so that satisfy $\pi(i_k) \equiv \pi(i_k \cup i_j), i_k \prec i_j, \forall i_j \in X_{lexcooc}$. Denoted as $lexcooc(i_k) = X_{lexcooc}$.

Definition 8. Let $i_k \in I(i_1 \prec i_2 \prec \dots \prec i_m)$ items are ordered in support ascending order, i_k is called a kernel item. Itemset $Y_{lexlooc} \subseteq I$ is called occurrence items with kernel item i_k in as least one transaction, but not co-occurrence items, so that satisfy $1 \leq |\pi(i_k \cup i_{lexlooc})| < |\pi(i_k)|, \forall i_{lexlooc} \in Y_{lexlooc}$. Denoted as $lexlooc(i_k) = Y_{lexlooc}$ (Fig. 3).

Additional command line 12, 13 and 14 into Algorithm 1:

```

12:   foreach  $i_k \in t_j$  do
13:       Kernel_COOC[k].cooc = lexcooc( $i_k$ )
14:       Kernel_COOC[k].looc = lexlooc( $i_k$ )

```

We have $looc(G) = \{B, D, E, F\}$, where $B, D \prec F \prec G \prec E$, so $lexlooc(G) = \{E\}$. Execute command line 12, 13 and 14 has result on Table 5.

Table 5. The Kernel_COOC array are co-occurrence items ordered in support ascending order.

Item	H	B	D	F	G	E	A	C
sup	1	2	2	3	5	7	8	8
cooc	E	A, C, E	A, C	A, C	A, C	∅	C	∅
looc	∅	G	F, G	G, E	E	A, C	∅	∅

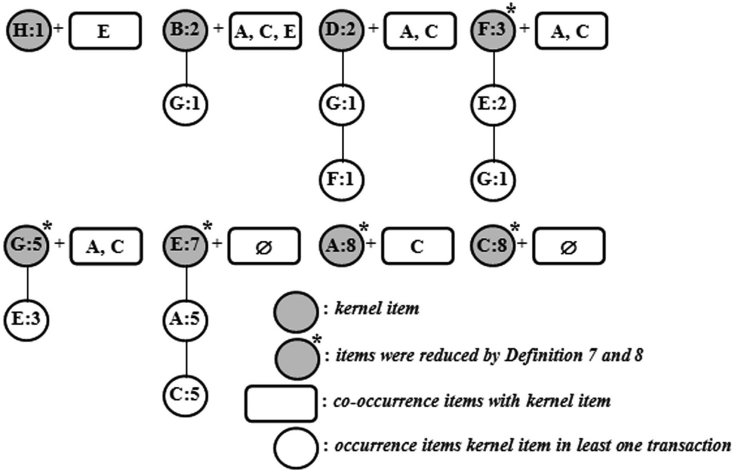


Fig. 3. The pattern-tree was reduced by Definitions 7 and 8.

3.2 Algorithm Generating All Frequent Itemsets

In this part, we illustrate the framework of the algorithm generating all frequent itemsets bases on the *Kernel_COOC* array.

Lemma 1. $\forall i_k \in I$, if $sup(i_k) \geq minsup$ and $X_{lexcooc}$ is powerset of $lexcooc(i_k)$ then $sup(i_k \cup x_{lexcooc}) \geq minsup, \forall x_{lexcooc} \in X_{lexcooc}$

Proof. According to Definition 8, (1) and (2): then $\pi(i_k) \equiv \pi(i_k \cup x_{lexcooc})$ and $sup(i_k) \geq minsup$. Therefore, we have $sup(i_k \cup x_{lexcooc}) \geq minsup$ ■.

Example 6. See Table 5. Consider the item *F* as kernel item ($minsup = 2$), we detect co-occurrence items with the item *F* as $lexcooc(F) = \{A, C\}$ and $X_{lexcooc} = \{A, C, AC\}$, then $sup(FA) = sup(FC) = sup(FAC) = 3 \geq minsup$.

Lemma 2. $\forall i_k \in I, Y_{lexlooc}$ is powerset of $lexlooc(i_k), \forall y_{lexlooc} \in Y_{lexlooc}$, if $sup(i_k \cup y_{lexlooc}) \geq minsup$ and $X_{lexcooc}$ is powerset of $lexcooc(i_k)$ then $sup(i_k \cup y_{lexlooc} \cup x_{lexcooc}) \geq minsup, \forall x_{lexcooc} \in X_{lexcooc}$.

Proof. According to Definitions 7, 8: then $|\pi(i_k \cup y_{lexlooc})| < |\pi(i_k)| = |\pi(i_k \cup x_{lexcooc})|$ and $sup(i_k \cup y_{lexlooc}) \geq minsup$. Therefore, we have $sup(i_k \cup y_{lexlooc} \cup x_{lexcooc}) \geq minsup, \forall x_{lexcooc} \in X_{lexcooc}, \forall y_{lexlooc} \in Y_{lexlooc}$ ■.

Example 7. See Table 5. Consider the item G as kernel item ($minsup = 2$), we detect co-occurrence items with item G as $lexcooc(G) = \{A, C\}$, $X_{lexcooc} = \{A, C, AC\}$; $lexlooc(G) = \{E\}$ and $sup(GE) = 3 \geq minsup$ then $sup(GEA) = sup(GEC) = sup(GEAC) = 3 \geq minsup$.

The framework of Algorithm 2 is presented as follows:

Algorithm 2. Generating all frequent itemsets satisfy $minsup$

Input : $minsup$, Kernel_COOC array, Dataset \mathcal{D}

Output: FI

- 1: **foreach** Kernel_COOC[k].sup $\geq minsup$ **do**
- 2: FI[k] = i_k
- 3: **if** (Kernel_COOC[k].sup = $minsup$) **then**
- 4: Co \leftarrow GenSub(Kernel_COOC[k].cooc)//generating noempty subsets of cooc
- 5: **foreach** $is_j \in Co$ **do**
- 6: FI[k] = FI[k] $\cup \{i_k \cup is_j\}$
- 7: **else**
- 8: **if** (Kernel_COOC[k].cooc = \emptyset) **then**
- 9: Lo \leftarrow GenSub(Kernel_COOC[k].looc)//generating noempty subsets of looc
- 10: **foreach** $is_j \in Lo$ **do**
- 11: FI[k] = FI[k] $\cup \{i_k \cup is_j\}$
- 12: **else**
- 13: Co \leftarrow GenSub(Kernel_COOC[k].cooc)
- 14: **foreach** $is_j \in Co$ **do**
- 15: $F_t = F_t \cup \{i_k \cup is_j\}$
- 16: Lo \leftarrow GenSub(Kernel_COOC[k].looc)
- 17: **foreach** $is_j \in Lo$ **do**
- 18: $F_k = F_k \cup \{i_k \cup is_j\}$
- 19: **foreach** $f_i \in F_t$ **do**
- 20: **foreach** $is_j \in Lo$ **do**
- 21: FI[k] = FI[k] $\cup \{f_i \cup is_j\}$
- 22: FI[k] = FI[k] $\cup F_t$
- 23: **sort** FI in descending by support

We illustrate Algorithm 2 on example database in Table 1, and $minsup = 3$. After the processing Algorithm 1, the Kernel_COOC array in Table 5 is showed.

Line 3, consider items satisfying $minsup$ as kernel items $\{F, G, E, A, C\}$;

Consider kernel item F , $sup(F) = 3 = minsup$ (Lemma 1- form line 5 to 6) generating all frequent with kernel item F as $FI_{[F]} = \{\underline{(F, 3)}, \underline{(FA, 3)}, \underline{(FC, 3)}, \underline{(FAC, 3)}\}$.

Consider the kernel item G (from line 12 to 21): the powerset of co-occurrence items of kernel item G as set $Co = \{A, C, AC\}$, generating frequent itemsets $F_t = \{\underline{(GA, 5)}, \underline{(GA, 5)}, \underline{(GAC, 5)}\}$; line 16 – generating noempty subsets of $looc$

field $Lo = \{E\}$, $F_k = \{GE\}$ – generating frequent itemsets $FI_{[G]} = \{(\underline{G}, 5), (\underline{GA}, 5), (\underline{GC}, 5), (\underline{GAC}, 5), (\underline{GE}, 3), (\underline{GEA}, 3), (\underline{GEC}, 3), (\underline{GEAC}, 3)\}$.

Consider the *kernel item E (from line 8 to 11)*: generating noempty subsets of *looc* field $Lo = \{A, C, AC\}$, line 10 and 11 – generating frequent itemsets $FI_{[E]} = \{(\underline{E}, 7), (\underline{EA}, 5), (\underline{EC}, 5), (\underline{EAC}, 5)\}$.

Consider the *kernel item A (similary kernel item G)*: $Co = \{C\}$, $F_t = \{(AC, 8)\}$, $Lo = \{\emptyset\}$, $F_k = \{\emptyset\}$ – generating frequent itemsets $FI_{[A]} = \{(A, 8), (AC, 8)\}$.

Consider the *kernel item C (similary kernel item E)*: $Lo = \{\emptyset\}$ – generating frequent itemsets $FI_{[C]} = \{(C, 8)\}$ (Fig. 4).

Table 6. All frequent itemsets satisfy $minsup = 3$ (example database in Table 1).

Kernel item	Frequent itemsets - FI							
F	(F, 3)	(FA, 3)	(FC, 3)	(FAC, 3)				
G	(GE, 3)	(GEA, 3)	(GEC, 3)	(GEAC, 3)	(GA, 5)	(GC, 5)	(GAC, 5)	(G, 5)
E	(EC, 5)	(EA, 5)	(EAC, 5)	(E, 7)				
A	(A, 8)	(AC, 8)						
C	(C, 8)							

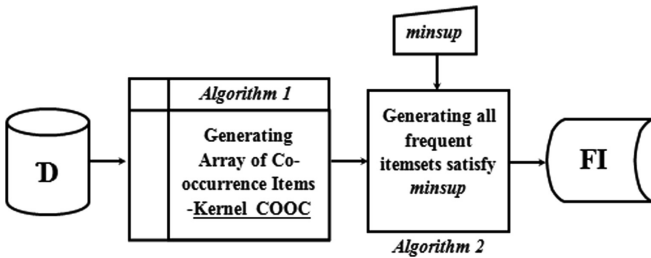


Fig. 4. The diagram sequential algorithm for frequent itemsets mining (SEQ-FI).

3.3 Parallel NPA-FI Algorithm Generating All Frequent Itemsets

In this section, we illustrate parallel algorithms and experimental setup on the multi-core processors (MCP). We proposed a parallel **NPA-FI** algorithm for because it quickly detects frequent itemsets on MCP using Algorithms 1 and 2.

The parallel **NPA-FI** algorithm for generating all frequent itemsets, including 2 phases:

- *Phase 1:* Computing Kernel_COOC array by parallelization Algorithm 1;
- *Phase 2:* Generating all frequent itemsets by parallelization Algorithm 2;

Phase 1 - Parallelization Algorithm 1 is shown in the diagram:

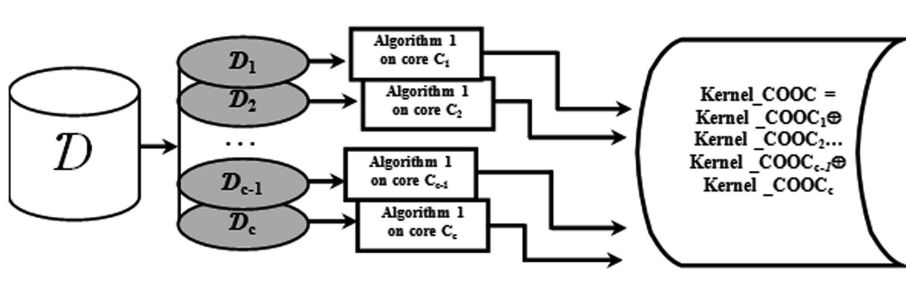


Fig. 5. The diagram parallelization phase 1.

In Fig. 5, we split the transaction database \mathcal{D} into c (number of core on CPU) parts $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_c$. After that, the core j^{th} executes Algorithm 1 with input transaction database \mathcal{D}_j , output the $Kernel_COOC_{\mathcal{D}_j}$ array. The $Kernel_COOC_{\mathcal{D}}$ array for the transaction database \mathcal{D} , we compute the following equation:

$$Kernel_COOC_{\mathcal{D}} = Kernel_COOC_{\mathcal{D}_1} \oplus Kernel_COOC_{\mathcal{D}_2} \oplus \dots \oplus Kernel_COOC_{\mathcal{D}_c} \tag{3}$$

\oplus denoted as **sum** for *sup*, **AND** for *cooc*, **OR** for *looc* field of each element array.

The next step, we sort the $Kernel_COOC$ array in ascending order by supporting, executing commands line 12, 13 and 14 of the Algorithm 1.

Example 8. See Table 1. We split the transaction database \mathcal{D} into 2 parts: the database \mathcal{D}_1 consists 5 transaction $\{t_1, t_2, t_3, t_4, t_5\}$ and database \mathcal{D}_2 consists 5 transaction $\{t_6, t_7, t_8, t_9, t_{10}\}$.

The processing of Algorithm 1 on database \mathcal{D}_1

Item	A	B	C	D	E	F	G	H
sup	4	0	4	1	3	2	3	1
cooc	10100000	11111111	10100000	10110110	00001000	10100100	10100010	00001001
looc	10111110	00000000	10111110	10110110	10101111	10111110	10111110	00001001

The processing of Algorithm 1 on database \mathcal{D}_2

Item	A	B	C	D	E	F	G	H
sup	4	2	4	1	4	1	2	0
cooc	10100000	11101000	10100000	10110000	00001000	10101110	10101010	11111111
looc	11111110	11101010	11111110	10110000	11101110	10101110	11101110	00000000

Results of Eq. (3), we have the $Kernel_COOC$ array as presented in Table 4.

Phase 2 – Parallelization of Algorithm 2 is shown in the diagram:

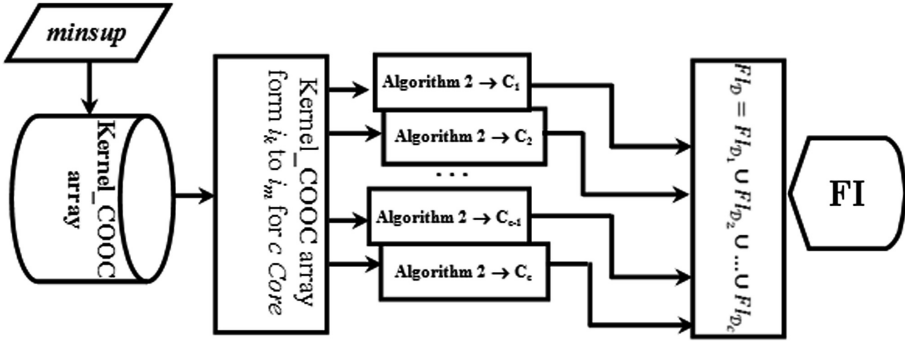


Fig. 6. The diagram parallelization phase 2.

In Fig. 6, we split the $Kernel_COOC_{\mathcal{D}}$ array from element i_k to i_m ($sup(i_k) \geq minsup$) into c parts. After that, the core j th execute Algorithm 2 with input $Kernel_COOC_{\mathcal{D}}$ array from $k + (j - 1) * ((m - k + 1) div c)$ to $k + j * ((m - k + 1) div c)$ element, returns results frequent itemsets $FI_{\mathcal{D}_j}$. The frequent itemsets $FI_{\mathcal{D}}$ for the transaction database \mathcal{D} , we compute the following equation:

$$FI_{\mathcal{D}} = FI_{\mathcal{D}_1} \cup FI_{\mathcal{D}_2} \cup \dots \cup FI_{\mathcal{D}_c} \tag{4}$$

Example 9. See Table 1. Generating all frequent itemsets satisfy $minsup = 3$, the transaction database \mathcal{D} split into 2 parts as Example 6. Results of phase 1 parallelization, we have the $Kernel_COOC_{\mathcal{D}}$ array as Table 5.

The processing of Algorithm 2 on the $Kernel_COOC$ array form item F to E:

Kernel item	Frequent itemsets - $FI_{\mathcal{D}_1}$							
F	(F, 3)	(FA, 3)	(FC, 3)	(FAC, 3)				
G	(GE, 3)	(GEA, 3)	(GEC, 3)	(GEAC, 3)	(GA, 5)	(GC, 5)	(GAC, 5)	(G, 5)
E	(EC, 5)	(EA, 5)	(EAC, 5)	(E, 7)				

The processing of Algorithm 2 on the $Kernel_COOC$ array form item A to C:

Kernel item	Frequent itemsets - $FI_{\mathcal{D}_2}$	
A	(A, 8)	(AC, 8)
C	(C, 8)	

Results of Eq. (4), we have all frequent itemsets as presented in Table 6.

4 Experiments

All experiments were conducted on a PC with a Core Duo CPU T2500 2.0 GHz (2 Cores, 2 Threads), 4 Gb main memory, running Microsoft Windows 7 Ultimate. All codes were compiled using C#, Microsoft Visual Studio 2010, .Net Framework 4.

We experimented on two instance types of datasets:

- Two real datasets that belong to *medium* instance are form of UCI Machine Learning Repository [<http://archive.ics.uci.edu/ml>] as **Pumsb** and **Retail** datasets.
- Two synthetic datasets that belong to *large* instance using the software are generated by IBM Almaden Research Center [<http://www.almaden.ibm.com>] as **T40I1KD100K** and **T40I1KD200K** datasets (Table 7).

Table 7. Datasets description in experiments.

Instance type	Name	#Trans	#Items	#Avg. length	Type
Medium	Pumsb	49,046	2,113	74	Dense
	Retail	88,162	16,470	10	Sparse
Large	T40I1KD100K	100,000	1,000	40	Sparse
	T40I1KD200K	200,000	1,000	40	Sparse

Deng et al. proposed the **PrePost** [9] algorithm for constructing a *FP-tree-like* and mining frequent itemsets from a database. In recent years, **PrePost** algorithm shows the better performance result. We have compared the parallel **NPA-FI** algorithm with sequential algorithms (**SEQ-FI**) and **PrePost** algorithm.

Performance implementation parallel **NPA-FI** algorithm on multi-core processors:

$$P = 1 - \left(T_M - \frac{T_S}{c} \right) / \left(\frac{T_S}{c} \right) \tag{5}$$

Where:

- T_S : executing time of the sequential algorithm;
- T_M : executing time of the parallel algorithm;
- c : number of the core on CPU.

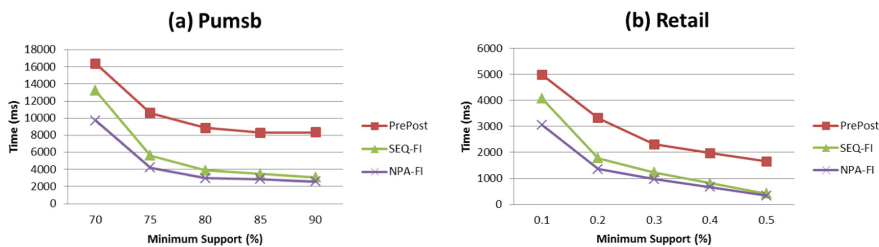


Fig. 7. Running time of the three algorithms on medium datasets.

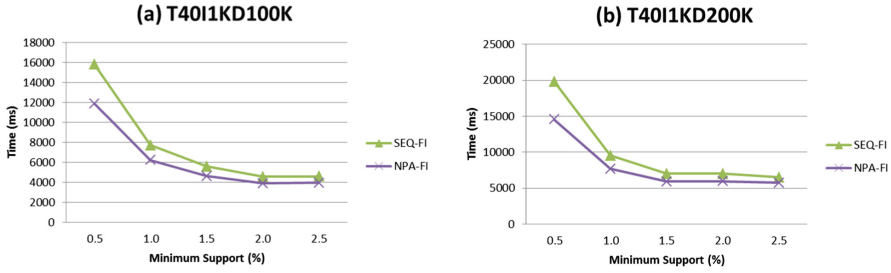


Fig. 8. Running time of the two algorithms on large datasets.

Figure 7(a) and (b) show the running time of the compared algorithms on medium datasets **Pumsb** and **Retail**. **SEQ-FI** runs faster **PrePost** algorithm under all minimum supports; **NPA-FI** runs faster **SEQ-FI** algorithm. Average performance of the parallel **NPA-FI** algorithm in turn: **Pumsb** as $\bar{P} = 0.78(78\%)$; $\sigma = 0.048$ and **Retail** as $\bar{P} = 0.79(79\%)$; $\sigma = 0.032$.

Figure 8(a) and (b) show the running time of the compared algorithms on large datasets **T40I1KD100K** and **T40I1KD200K**. **PrePost** algorithm fails to frequent itemsets mining on *large datasets*; **NPA-FI** runs faster **SEQ-FI** algorithm. Average performance of the parallel **NPA-FI** algorithm in turn: **T40I1KD100K** as $\bar{P} = 0.81(81\%)$; $\sigma = 0.045$ and **T40I1KD200K** as $\bar{P} = 0.81(81\%)$; $\sigma = 0.052$.

In summary, experimental results suggest the following ordering of these algorithms as running time is concerned: **SEQ-FI** runs faster **PrePost** algorithm; **NPA-FI** runs faster **SEQ-FI** algorithm. Average performance of the parallel **NPA-FI** algorithm on datasets experimental is $\bar{P} = 0.80(80\%)$; $\sigma = 0.042$.

5 Conclusion

In this paper, we have proposed a sequential architecture mining frequent itemsets on large transaction databases, consisting of two phases: *the first phase*, quickly detect a the **Kernel_COOC** array of co-occurrences and occurrences of kernel item in at least one transaction; *the second phase*, the algorithm is proposed for fast mining all frequent itemset based on **Kernel_COOC** array. Besides, when using mining frequent itemsets with *other minsup* value then the proposed algorithm only performs mining frequent itemsets based on the **Kernel_COOC** array that is calculated previously, reducing the significant processing time. The next step, we develop a sequential algorithm for mining frequent itemsets and thus parallelize the sequential algorithm to effectively demonstrate the multi-core processors. The experimental results show that the proposed algorithms perform better than other existing algorithms.

The results from the algorithm proposed: In the future, we will expand the algorithm to be able to mining frequent itemsets on weighted transaction databases, as well as to expand the parallel **NPA-FI** algorithm on distributed computing systems such as Hadoop, Spark.

Acknowledgements. This work was supported by University of Social Sciences and Humanities; University of Science, VNU-HCM, Vietnam.

References

1. Agrawal, R., Shafer, J.: Parallel mining of association rules. *IEEE Trans. Knowl. Data Eng.* **8**, 962–969 (1996)
2. Dong, J., Han, M.: BitTableFI: an efficient mining frequent itemsets algorithm. *Knowl. Based Syst.* **20**(4), 329–335 (2007)
3. Song, W., Yang, B.: Index-BitTableFI: an improved algorithm for mining frequent itemsets. *Knowl. Based Syst.* **21**, 507–513 (2008)
4. Fournier-Viger, P., Lin, J.C.-W., Vo, B., Chi, T.T., Zhang, J., Le, H.B.: A survey of itemset mining. *WIREs Data Min. Knowl. Discov.* **7**(4), e1207 (2017). <https://doi.org/10.1002/widm.1207>
5. Agrawal, R., Imilienski, T., Swami, A.: Mining association rules between sets of large databases. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Washington, DC, pp. 207–216 (1993)
6. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: *Proceedings of the 2000 ACM-SIGMOD International Conference Management of Data (SIGMOD 2000)*, Dallas, TX, pp. 1–12 (2000)
7. Lin, M.Y., Lee, P.Y., Hsueh, S.C.: Apriori-based frequent itemset mining algorithms on MapReduce. In: *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication*, New York, NY, USA, p. 76 (2012)
8. Moonesinghe, H.D.K., Chung, M.J., Tan, P.N.: Fast parallel mining of frequent itemsets. Technical report no. 2, Department of Computer Science and Engineering, Michigan State University (2006)
9. Deng, Z.H., Wang, Z.H., Jiang, J.J.: A new algorithm for fast mining frequent itemsets using N-lists. *Sci. China Inf. Sci.* **55**(9), 2008–2030 (2012)
10. Djenouri, Y., Bendjoudi, A., Djenouri, D., Habbas, Z.: Parallel BSO algorithm for association rules mining using master/worker paradigm. In: Wyrzykowski, R., Deelman, E., Dongarra, J., Karczewski, K., Kitowski, J., Wiatr, K. (eds.) *PPAM 2015. LNCS*, vol. 9573, pp. 258–268. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-32149-3_25



A Geo-Tagging Framework for Address Extraction from Web Pages

Julia Efremova^(✉), Ian Endres, Isaac Vidas, and Ofer Melnik

HERE Technologies, Amsterdam, The Netherlands
{julia.efremova,ian.endres,isaac.vidas,ofer.melnik}@here.com

Abstract. Searching for locations in web data and associating a document with a corresponding place on the map becomes popular in user's daily activities and it is the first step in web page processing. People often manually search for locations on a web page and then use map services to highlight them because geographic information is not always explicitly available.

In this work, we present a geo-tagging framework to extract all addresses from web pages. The solution includes an efficient web page processing approach, which combines a probabilistic language model with real-world knowledge of addresses on maps and extends geocoding services from short queries to large text documents and web pages. We discuss the main problems in dealing with web pages such as: web page noise, identification of relevant segments, and extraction of incomplete addresses. The experimental result shows precision above 91% which outperforms standard baselines.

1 Introduction

Web pages and text documents often contain geographical information mentioned as free text. We consider under geographic information the location of a real place, for instance a theater, a shop or a restaurant. Locations typically occur in a document in the form of unstructured descriptions instead of specified geo-coordinates with a point on the map and they are difficult to identify compared to structured data.

The importance of location extraction is the possibility to associate them with the exact map coordinates: latitude and longitude. This process is called geocoding. Available geo-information increases the popularity of a place and can be used as a part of an advanced content analysis. Locations are widely used by recommendation systems which often display the most relevant places with specifically mentioned locations and ignore free text.

In this work, we focus on geo-tagging of web pages and our goal is to identify locations in a document and to assign them appropriate geotags. Geo-tagging is common in social media, for instance, Twitter posts, Flickr pictures, etc. often contain geotags.

Dealing with web pages is different from identifying locations in structured documents and in short text queries, due to web page heterogeneity, lack of

structure and a high level of noise. They require special pre-processing techniques to reduce the amount of irrelevant information and to discover targeted parts.

In this work, we design a geo-tagging framework for address extraction from web pages. The main distinction of our work is a designed framework for full text geocoding, which combines state-of-the-art techniques from various fields (web page processing, information extraction and machine learning), extends standard address extraction solutions to the domain of web pages.

The contribution of this paper is a designed framework for address extraction which has the following advantages:

1. highlighting of the main aspects of web page processing such as: web page segmentation and noise reduction;
2. augmenting a probabilistic language based model with geographic information;
3. validating addresses by real-world knowledge of maps on top of the process;
4. resolving similar and partial addresses (address resolution) by ranking results;
5. extending address extraction from short queries large text documents and web pages.

2 Related Work on Address Extraction from Web Pages

The problem of address extraction has been studied by various researchers. Recently, many approaches to address extraction from free text have been proposed. Most of them belong to the natural language processing fields and include techniques from sentence breaking and part-of-speech tagging to content parsing and semantic analysis [3].

Yu [10] presented address extraction using rule-based, machine learning and hybrid techniques based on decision tree classifier and n -gram model. He uses regular expressions combined with lexical features such as: punctuation, initial capital, contain digits, etc.

Chang and Li [2] proposed the MapMarker framework where they adopt conditional random fields (CRF) for web page segmentation and solve address extraction problem by sequence labeling. Their approach relies on the quality of training data and the learned language model.

Melo and Martins [7] prepared a survey where they predict the geo-coordinates for a document using the whole textual content. They show early approaches based on rule-based heuristics; machine learning approaches including feature selection and classification; and extended approaches using place disambiguation and coordinate estimations. The difference of our approach in assumption that a given document can contain multiple locations and we verify identified addresses by searching them on the real map.

Another existing solution for address parsing and expansion is Libpostal¹ which is an open source library trained on a large dataset of real-world addresses

¹ <https://mapzen.com/blog/libpostal>.

from OpenStreetMap² [4,5]. Libpostal parses addresses by searching for house numbers, roads, cities, etc. For our purposes, the main shortcoming of Libpostal is the lack verification whether an address really exists on the map without identification of empty requests.

The distinction of our work comparing to prior efforts is in applying geocoding service on top of the whole extraction chain and use it as the main address validator. Another distinction is in address resolution by ranking similar addresses in the same neighborhood and final identification of complete and verifies addresses.

3 Data Description: A Collection of Web Pages

We use a collection of crawled and cached web pages provided by *Common Crawl* [8]. Every web page is available in the WARC, WAT and WET formats which stand for raw web page data, metadata and extracted text. Common Crawl is a public corpus, mostly stored on Amazon Web Services³.

A subset of the CommonCrawl dataset has schema information in the microdata format with manual annotation of the main content such as: addresses, names, phones, as well as micro content such as: streets, regions, postal codes, etc. Microdata are a set of attributes which are embedded into standard HTML tags as a special property *itemprop*.

Table 1. An example of microdata annotation. The annotated content is in bold

```

<DIV class="column threeColumn">
<DIV itemprop="address" itemtype="http://schema.org/PostalAddress">
<SPAN itemprop="streetAddress">505 Grand Rd</SPAN>
<BR/><SPAN itemprop="addressLocality">East Wenatchee</SPAN><BR/>
<SPAN itemprop="addressRegion">WA</SPAN>
<SPAN itemprop="postalCode">98802</SPAN><BR/></DIV>
<DIV><A href="/restaurateurs/your-business/6023/">
Is this your business?</A></DIV>

```

Table 1 is an example of detailed address annotation (street, locality, region and postal code) embedded into the DIV and SPAN tags. Documents with microdata annotation are a great source for training and evaluation purposes. According to the W3Techs survey only around 12.7% web pages have microdata tags compared to the entire web; hence, there is a need for automatic extraction. Microdata is an excellent source of ground truth, which can be used for training and evaluation purposes. It provides structured data and makes web pages easier to read instead of dealing with free text. Particularly, we use the *Restaurant* schema⁴ from Schema.org of microdata annotations.

² <https://osmnames.org>.

³ <https://aws.amazon.com/public-datasets/common-crawl/>.

⁴ <http://schema.org/Restaurant>.

4 An Introduced Geo-Tagging Framework

Our address extraction framework includes the following steps: *text extraction*, *tokenization*, *annotation by geographical features*, *annotation by lexical features*, *sliding window generation*, *window classification*, *window geocoding* and *address ranking* illustrated in Fig. 1.

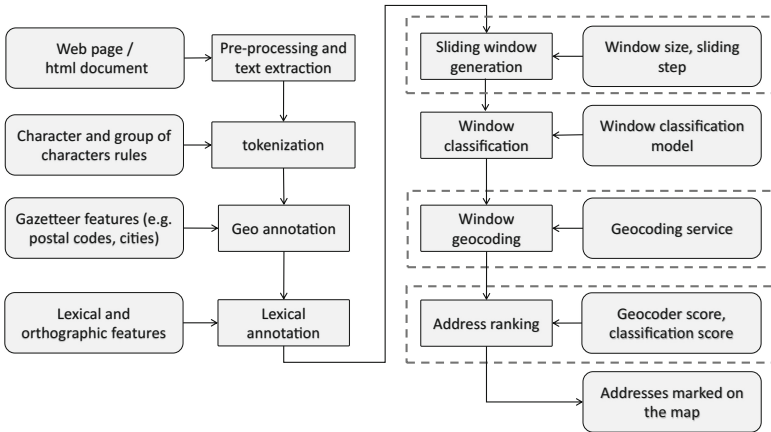


Fig. 1. Components of an introduced address extraction framework

In the beginning, a web document is cleaned and preprocessed in order to reduce HTML noise and to convert a raw document into an informative representation.

Then in the *tokenization* step, data is divided into small elements called tokens (words, bi-grams, tri-grams, etc.) based on a certain splitting criteria.

In the third and the fourth step, we annotate tokens by different groups of features: lexical described in [10] which indicate whether a token is a digit, contains a digit, is a title, is capitalized, starts with a capital letter; and geographical features, which show if a token is a state, is a postal code, is a city, etc. The *Geonames*⁵ [1] project and the *Libpostal* library discussed in Sect. 2 can be used as a source of geo-features.

Subsequently, term frequency transforms feature annotation into numerical feature vectors. It calculates the number of times each feature occurred.

Afterwards, a sliding window of a fixed size and with a fixed sliding step moves over annotated tokens from top to bottom. A approach based on sliding windows allows to analyze the entire web page content and does not require prior web page segmentation or other pre-processing. A common solution for this step is web page segmentation, for instance by Latent Semantic Indexing [6]. LSA identifies semantically close segments by associating related words, however it

⁵ <http://www.geonames.org>.

leads to missing addresses due to incorrect segments which we aim to avoid. Table 2 illustrates sliding windows for the piece of text ‘Papa Murphy’s East Wenatchee 505 B Grant Road East Wenatchee WA, 98802’ in the example in Table 1.

Table 2. An illustration of sliding windows

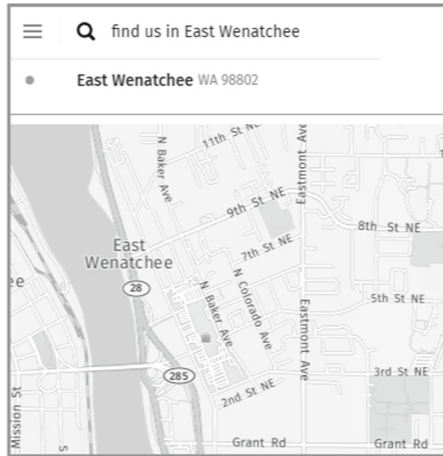
w_1	Papa Murphy’s East Wenatchee 505 B Grant Road East Wenatchee
w_2	Murphy’s East Wenatchee 505 B Grant Road East Wenatchee WA
w_3	East Wenatchee 505 B Grant Road East Wenatchee WA, 98802

After creating a sliding window, a binary classifier computes a classification score for each window and identifies address candidates. We choose a *Support Vector Machines* (SVM) classifier, since it is a widely-used robust classifier [9]. Classification requires a prior training phase on an annotated dataset to make a prediction on new data. We consider annotated addresses in the dataset discussed in Sect. 3 as positive examples and windows without addresses as negative.

Then, the Geocoder validates a candidate window by searching its geo-coordinates on the map. We use HERE Geocoder⁶ which provides a powerful service for address parsing and validation. Figure 2a illustrates address validation by searching it on the map.



(a) an example of a complete address search



(b) an example of a partial address search

Fig. 2. An illustration of address verification by searching it on the map

An exact point location could not be identified on the map in case of partial addresses. Geocoder then refers to a general area as shown in Fig. 2b.

⁶ <https://developer.here.com>.

The last step is addresses ranking to solve the address disambiguation problem. A web page can contain the same address mentioned multiple times, address variations and parts of the same address. As an example consider the two addresses: ‘505 Grand Rd East Wenatchee’ and ‘East Wenatchee’, the second refers to a region of the first place. We use a Geocoder relevance score and a classification score to solve the problem of similar addresses and to find final and the most detailed and verified addresses on a web page.

5 Experimental Results

5.1 Applying a Geo-Tagging Framework

Before applying the designed framework, it is important to define a window size and a sliding step based to the two main criteria:

1. a window size should fit a complete address;
2. a window size should have a minimum number of irrelevant tokens (noise);
3. a sliding step should not be large otherwise parts of the same address will belong only to neighboring windows.

We learn an optimal window size from length distributions of annotated addresses presented on Fig. 3. The typical address length is between 5 to 10 tokens in microdata annotation and between 6 and 10. We choose the window size of 10 tokens which is an optimal size for most of the addresses.

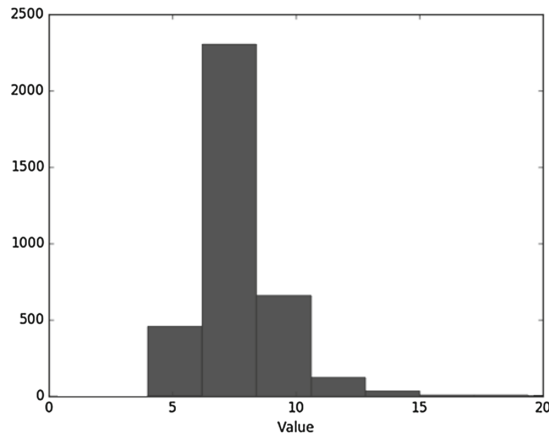


Fig. 3. Distribution of address length in tokens. X-axis represents the address length and Y-axis represents the number of addresses analyzed

5.2 Result Evaluation

We conduct experiments on the annotated datasets described in Sect. 3. In order to assess the performance of our results, we apply 10-fold cross-validation. We randomly partition the annotated data into 10 equal size subsets. Then one subset is chosen as the validation data for testing the classifier, and the remaining subsets are used for training purposes. Then the process is repeated 10 times, with each of the 10 subsets used exactly once as the validation dataset.

We use regular expressions (RE) as the first baseline and learn the RE grammar on a training set considering, for instance text between a house number (represented by digits) and a postal code as an address.

The second baseline is based on segmentation by key words. The Latent Semantic Analysis (LSA) introduced in Sect. 4 searches for relevant segments similar to a list of keywords obtained by topic modeling or from a relevant vocabulary. In our case, we use GeoNames database as a source of initial geo-keywords.

When segments are identified, we search for the full address only within the identified segments. This method assumes that an address should contain keywords (e.g. cities, postal codes) and it is not robust in case of misspellings, for instance comparing ‘*San Francisco*’ and ‘*SF*’.

Table 3 shows comparative evaluation of the designed geo-tagging framework and two baselines in standard metrics: precision and recall. Address extraction by RE has the lowest performance since addresses can occur in any format which is differ from RE. The baseline 2 demonstrates almost double improvement in performance however recall is only 50% due to missed address segments. The proposed approach outperforms the baselines in terms of precision and also recall which is improved up to around 92%.

Table 3. Evaluation of the geo-tagging framework for address extraction

Address extraction	Precision	Recall
Baseline 1: regular expressions	47.7%	33.61%
Baseline 2: LSA segmentation with subsequent classification	80.67%	53.61%
The proposed geo-tagging framework	91.12%	92.86%

Initial error analysis of incorrectly extracted addresses revealed common issues. One of which is confusions between the main address elements, for instance place names and street names, house numbers and other numbers in a window. If a number (which is not an address part) occurs in a window, it can be matched it to a house number and affect address extraction.

Therefore, identification of precise address boundaries can be a potential extension of this work. One solution is to learn an address grammar. For instance, a postal code can be used as an indicator of the ending boundary.

Probabilistic approaches based on sequence annotation such as conditional random fields is another way to find address boundaries, however it might reduce recall and lead to missing addresses because it does not take into account real map knowledge.

6 Conclusion

In this work, we introduced a geo-tagging framework for address extraction from web pages. We addressed the main challenges in working with web data such as: structural heterogeneity, web page pre-processing, segment analysis and address validation. We incorporated sliding windows and avoided web page segmentation in order to improve recall and to save precision on the high level. We discussed in detail different groups of feature extraction and also sources to obtain geo-information. We showed how to validate extracted addresses by analyzing their existence on the map and applied geocoding on the top of machine learning models.

It is possible to extend the designed framework in various directions, for instance by adding visual features, by incorporating the DOM structure or defining a window size dynamically instead of having it fixed.

References

1. Ahlers, D.: Assessment of the accuracy of geonames gazetteer data. In: Proceedings of the 7th Workshop on Geographic Information Retrieval, GIR 2013, pp. 74–81. ACM, USA (2013)
2. Chang, C.-H., Li, S.-Y.: MapMarker: extraction of postal addresses and associated information for general web pages, pp. 105–111. IEEE Computer Society (2010)
3. Gupta, V., Lehal, G.S.: A survey of text mining techniques and applications. *J. Emerg. Technol. Web. Intell.* **1**(1), 60–69 (2009)
4. Haklay, M., Weber, P.: Openstreetmap: user-generated street maps. *Pervasive Comput.* **7**(4), 12–18 (2008)
5. Lawrence, C., Riezler, S.: NLmaps: a natural language interface to query OpenStreetMap. In: COLING, Demos, pp. 6–10. ACL (2016)
6. Li, H., Xu, J.: Semantic matching in search. *Found. Trends Inf. Retr.* **7**(5), 343–469 (2014)
7. Melo, F., Martins, B.: Automated geocoding of textual documents: a survey of current approaches. *Trans. GIS* **21**(1), 3–38 (2017)
8. Meusel, R., Petrovski, P., Bizer, C.: The webdatacommons microdata, RDFa and microformat dataset series. In: Mika, P., et al. (eds.) ISWC 2014. LNCS, vol. 8796, pp. 277–292. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11964-9_18
9. Witten, I.H., Frank, E., Hall, M.A.: *Data Mining: Practical Machine Learning Tools and Techniques*, 4th edn. Morgan Kaufmann Publishers Inc., Burlington (2016)
10. Yu, Z.: High accuracy postal address extraction from web pages (2007)



Data Mining for Municipal Financial Distress Prediction

David Alaminos^{1(✉)}, Sergio M. Fernández^{2(✉)}, Francisca García^{3(✉)},
and Manuel A. Fernández^{1(✉)}

¹ Department of Finance and Accounting, University of Malaga, Malaga, Spain
{alaminos, mangel}@uma.es

² Department of Languages and Computer Sciences, University of Malaga,
Malaga, Spain
sergiofernandezmiguelez@uma.es

³ Department of Applied Economics, University of Malaga, Malaga, Spain
fg_lopera@uma.es

Abstract. Data mining techniques are capable of extracting valuable knowledge from large and variable databases. This work proposes a data mining method for municipal financial distress prediction. Using a new proxy of municipal financial situation and a sample of 128 Spanish municipalities, the empirical experiment obtained satisfactory results, which testifies to the viability and validity of the data mining method proposed for municipal financial distress prediction.

Keywords: Financial distress prediction · Data mining · Municipalities
Individual classifiers · Local governments

1 Introduction

Municipal financial distress is a global phenomenon which has captured the attention of researchers and managers of public institutions with the aim of contributing to the continuity and financial sustainability of public services. After the last financial crisis, concerns about the debts incurred by municipalities have increased significantly. Municipal debts are particularly worrying because they affect not only the daily life of citizens but also local private companies, who depend on public decisions [1]. In this context, numerous models have been developed to assess municipal financial distress [2–4]. These models can be divided into two types [5]. On the one hand, the models which analyse the financial situation each year to determine if there exists a state of emergency. On the other hand, the models which analyse fiscal capacity by using tendencies to predict revenues and spending [6].

Although the models for assessing municipal financial distress have achieved significant success, one of the main limitations thereof is related to measurement of financial condition, since it is not directly observed and the researchers have created an index following previous work [7, 8]. For this reason, current literature demands new research which will permit a comparison of results using others proxies of the financial situation [9]. With the objective of covering this gap, this work proposes a model for

assessing municipal financial distress which incorporates a new proxy of financial situation. This new proxy is the criteria used by Spanish legislation, which refers to the ratio of default to municipal commercial debt. To that end, this work applied a data mining focus to a sample of Spanish municipalities, and this enabled their level of financial distress to be rated convincingly using a set of variables corresponding to 2015.

The work is organised in the following way. After the introduction, the Sect. 2 offers a review of the literature relating to municipal financial distress prediction. The Sect. 3 presents the model proposed and the different methodologies used. The Sect. 4 is devoted to data collection and variables. In the Sect. 5 we present the empirical results obtained. And lastly, in the Sect. 6, we present the main conclusions and implications.

2 Literature Review

There is abundant research into municipal financial distress and it is characterised by the differing approaches and methods of analysis used. Initially, statistical techniques of logistic regression and discriminant analysis were used, and more recently, methods based on heuristic approaches, such as multi-criteria methods [14, 17]. For their part, the explanatory variables used have been financial ratios of liquidity, solvency, efficiency and activity and, in other cases, political variables such as fiscal decentralisation or even socio-economic variables such as the level of unemployment, population or quality of infrastructure, varying according to the financial proxy used. The first works to address municipal financial distress were carried out in countries such as the United States of America and Australia, where the availability of information was more advanced than in other developed countries. [10] carried out a study on the prediction and prevention of fiscal crises in local governments in the United States, concluding that less than half the cases attempted to predict municipal financial distress. The analysis determined that the lack of preparation on the part of politicians, business failures, demographic changes and the increase in the cost of public services were the causes of the financial crisis. For their part, [4] applied a model of nine indicators to a sample of municipalities in the state of Michigan, grading their fiscal distress on a scale of 0–10 points, and highlighting the importance of revenue growth and the covering of costs in the budget. Later, [11] developed a statistical model to explain financial distress in Australian municipalities. They concluded that the degree of financial distress is positively associated with the size of their population and the nature of their revenues. [12] analysed municipal financial distress in the United States by addressing four dimensions (cash solvency, budgetary solvency, long-term solvency and services), the socio-economic environment and the size of government, with a binary dependent variable based on payment difficulties. The results led to the conclusion that budgetary surplus, short-term debt and the composition of sources of revenue are essential factors when it comes to determining municipal financial distress.

Ever since European countries started to compile and publish financial details of their municipalities, additional research has also been carried out with reference to the European continent. For example, [13] analysed the unequal implementation of municipal accounting systems in the European Union and the reasons why some

countries have resisted carrying out the necessary reforms. [14] constructed a model for predicting financial distress in Greek municipalities using a multi-criteria methodology. They obtained a level of accuracy which varied from 64.7% to 75.9% in the classification of bankrupt municipalities, and close to 100% for solvent municipalities. For their part, [2] analysed financial performance in Irish municipalities with a benchmarking methodology, concluding that financial autonomy and commercial debt are the most important variables. [15] used logistic regression to study the financial cost of local governments in Spain, calculating the risk premium of the municipalities with a multi-state dependent variable according to the reason for default, which enabled them to predict their possible state of insolvency. With this model, they obtained classifications with a level of accuracy of 76%, showing short-term debt, per capita income and the political ideology of the local government to be significant factors. Furthermore, [16] analysed municipal financial distress in Spain, taking into account three aspects of municipal finances (debt, revenues and services). Their results suggest that municipal revenues are the most significant aspect. [17] developed a multi-criteria model to evaluate the financial performance of French municipalities and reached the same conclusion regarding the importance of own revenues and operating expenses. Recently, [3] studied the municipal financial distress in Italy using a set of financial indicators and the logistic regression methodology. Their work, with a level of accuracy of 75%, reveals that staff costs relative to revenues, the rotation of short-term debts relative to revenues, and the level of dependency on subsidies are good predictors of municipal financial distress. Finally, [18] examined the factors which influence municipal credit risk, using the case of municipal default as a dependent variable. They found that the size of the population, per capita income and the composition of the debt determine the probability of possible municipal financial distress. Their results achieved a level of accuracy of 69.14% using financial variables and 79.73% when socio-economic variables were also included.

3 Data Mining Method for Municipal Financial Distress Prediction

Data mining is the process of extracting knowledge from databases and other information storage media. It has several functions, such as association, classification and prediction analysis, to name but a few. Each of them can use various alternative data mining algorithms [19]. Data mining for municipal financial distress prediction needs five steps: creating sample, data preprocessing, construction of model, accuracy assessment, and classification and prediction, as shown in Fig. 1. Creating sample means obtaining the relevant data from the sources which provide financial information about municipalities. Data preprocessing consists of the discretization of attributes of continuous values, data generalisation, attribute relativity analysis, and the elimination of outlying values. Construction of model refers to learning inductively from the data preprocessed by the algorithms used and choosing the model which best represents the classification of knowledge for municipal financial distress prediction. Accuracy assessment is the task of evaluating the model's predictive accuracy by means of the set of training data and the set of validating data. For each of the selected individual

classifiers, it has been tested with different sets of independent variables, in order to find out which of them gives a better classification performance. We randomly selected 500 sets of variables, to which 10-fold cross-validation was applied, randomly dividing the available set of samples by 80% for training samples, and 20% for test samples. We chose the set of variables that yielded the maximum of classification hits on the validation set and we have offered results according to the number of average hits on the test set (again averaged over the 10-fold cross-validation that has been made for each set). Finally, classification and prediction consists of using the model developed to predict municipal financial distress.

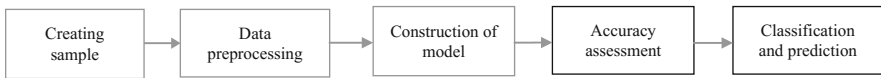


Fig. 1. Data mining steps in municipal financial distress prediction

Data mining which focusses on the prediction of municipal financial distress basically resolves a problem of classification and prediction. For that reason, in this work we use the classifiers which have obtained the best results in previous literature in models of financial distress prediction, specifically, Decision Trees, Naive Bayes, Multi-layer Perceptron, and Deep Belief Network [20].

A Decision Tree (DT) is a graphical, analytical form for classifying data by means of different possible paths [21]. Each of the nodes of the tree represents the different attributes of the data, the branches of the tree represent the possible paths to follow in order to predict the class of a new example, and the end nodes or leaves establish the class to which the test example belongs if the path along the branch in question is followed. The language of description of the DTs corresponds to the formulae in DNF (Disjunctive Normal Form). Thus, and in the event of having 3 attributes available (A, B and C), each of them with two values, x_i and $\neg x_i$, where $i = 1, 2, 3$, 2^n combinations can be constructed in CNF (Conjunctive Normal Form). Each of the combinations in CNF describes a part of the tree, and thus we would have disjunctives of the form expressed in (1).

$$(x_2 \wedge \neg x_3) \vee (x_2 \wedge x_3) \vee (\neg x_2 \wedge x_1) \vee (\neg x_2 \wedge \neg x_1) \quad (1)$$

These disjunctives are descriptors of the tree constructed, and 2^{2n} possible descriptions could be formed in DNF. As the order of the DT is very large, it is not possible to explore all of the descriptors to see which is the most suitable, and therefore heuristic search techniques are used to find a simple and quick way of doing so. The majority of DT construction algorithms are based on the Hill Climbing strategy, used in Artificial Intelligence to find the maximums or minimums of a function by means of a local search. This is an algorithm which begins with an empty tree and then segments into sets of examples, choosing in each case that attribute which best discriminates between the classes, until the tree is complete. In order to know which attribute is the

best, a heuristic function is used, and the choice is irrevocable, and it is therefore essential to ensure that the attribute in question is the closest to the optimum.

For its part, the Naive Bayes (NB) classifier is considered as part of the probabilistic classifiers, which are based on the supposition that quantities of interest are governed by probability distributions, and that the optimum decision can be taken by reasoning about these probabilities together with the data observed. In classification tasks, this algorithm is among the most used [22, 23]. [24] presents a basic guide to the different directions taken by research into NB in accordance with the modifications made to the algorithm. In this work we employ the traditional NB. The NB classifier is constructed using a T_r to estimate the probability of each class. In this way, when a new instance is presented, the classifier assigns to it the most probable category, which will be that which fulfils Eq. (2), i.e. the probability that, once the values which describe this new instance are known, it belongs to class C_i (which is the value of classification function $f(x)$ in finite set C).

$$c = \arg \max_{c_i \in C} P(c_i | i_j) \quad (2)$$

Using the Bayes theorem to estimate probability, we obtain Eq. (3).

$$c = \arg \max_{c_i \in C} \frac{P(c_i | i_j) P(c_i)}{P(i_j)} \quad (3)$$

In Eq. (3) the denominator does not differ between categories and can therefore be omitted, giving us (4).

$$c = \arg \max_{c_i \in C} P(c_i | i_j) P(c_i) \quad (4)$$

If we also resort to the hypothesis of conditional independence, i.e. to the assumption of independence between attributes, it is then possible to assume that the characteristics are conditionally independent given the classes. This simplifies the calculations, resulting in Eq. (5)

$$c = \arg \max_{c_i \in C} P(c_i) \prod_{k=1}^n P(a_{kj} | c_i) \quad (5)$$

where $P(C_i)$ is the fraction of examples in T_r which belong to class C_i , and $P(a_{kj} | C_i)$ is calculated in accordance with Bayes' theorem.

The so-called Multi-Layer Perceptron (MLP) is a supervised neural network model, which would be composed of a layer of inputs (sensors), another output layer, and a given number of intermediate layers, called hidden layers in that they have no connections to the exterior. Each input sensor would be connected to the units in the second layer, these in turn to those of the third layer, and so on. The aim of the network is to establish a correspondence between a set of inputs and a set of desired outputs. [25] confirmed that learning in MLP constituted a special case of functional approximation, where no assumption exists regarding the model underlying the data analysed. The learning process means finding a function which correctly represents the learning patterns as well as carrying out a process of generalisation which permits the efficient

treatment of individuals not analysed during learning [26]. To do this, weights W are adjusted on the basis of the information deriving from the sample set, considering that both the architecture and the network connections are known, and with the aim of obtaining those weights which will minimize learning error.

Thus, given a set of pairs of learning patterns $\{(x_1, y_1), (x_2, y_2) \dots (x_p, y_p)\}$ and a function of error $\varepsilon (W, X, Y)$, the training process entails the search for the set of weights which minimizes learning error $E(W)$ [26], as expressed by Eq. (6).

$$\min_w E(W) = \min_w \sum_{i=1}^p \varepsilon(W, x_i, y_i) \tag{6}$$

The majority of the analytical models used to minimize the function of error employ methods which require the evaluation of the local gradient of function $E(W)$, though techniques based on second order derivatives may also be considered [27]. While this is an area in constant development, the learning algorithms for the more common MLP-type networks are the backpropagation algorithm and its different variables, algorithms based on the conjugate gradient and the quasi-Newton models.

Lastly, Deep Belief Network (DBN) is a class of deep neural network where the two upper layers are modelled as an unsupervised bipartite associative memory. For their part, the lower layers of the network constitute a supervised graphical model called a sigmoid belief network. The difference between sigmoid belief networks and DBN lies in the parametrization of the hidden layers [26].

Equation (7) describes a DBN, where v is the vector of visible units, $P(h^{k-1}|h^k)$ is the conditional probability of visible units given the hidden ones at level k , and $P(h^{l-1}, h)$ is the joint distribution at the top level for $x(n) = [1, x_1(n), x_2(n), \dots, x_m(n)]^T$

$$P(v, h^1, \dots, h^l) = P(h^{l-1}, h^l) \left(\prod_{k=0}^{l-2} P(h^k|h^{k+1}) \right) \tag{7}$$

When we apply DBN to a set of data, we are looking for a model $Q(h^{l-1}, h)$ for the true posterior $P(h^{l-1}, h)$. The subsequent Q s are all approximations, except for the top level $Q(h^{l-1}, h)$ which is equal to the true posterior $P(h^{l-1}, h)$ and allows us to make an exact inference [26].

4 Data Collection and Variables

This work uses a sample of 128 Spanish municipalities. All fulfil the condition of having a population of over 50,000 inhabitants, in line with the criteria proposed by [15]. The information corresponding to the municipalities in the sample refers to 2015 and is provided by the Spanish Court of Auditors, which publishes data related to the annual accounts of Spanish municipalities. In addition, and in order to validate the models to be estimated and to test predictive ability, test samples were used that were different and unrelated to those used in estimating the model. We then proceeded to

divide the data into two different samples, one used to build the model (training data) and another for testing it (testing data).

The majority of the studies which address municipal financial distress have used explanatory variables which refer to the municipalities' financing structure. For this study, we have selected a set of financial variables from among those most used in the previous literature [14, 15]. These selected variables comprehensively reflect the financial structure of Spanish municipalities and are related to indebtedness, municipal revenues, the capacity of revenues to pay debts and cover costs, and the volume of subsidies. Moreover, we use variables of political transparency which indicate the quality of information and management of a municipality [18, 28]. Table 1 shows the definition of the variables used in the study.

Table 1. Variables definition

Code	Description
<i>Financial variables</i>	
F1	Financial debt/commercial debt
F2	Own revenue/total revenue
F3	Short-term debt/long-term debt
F4	Total debts/total assets
F5	Own revenue/total debts
F6	Short-term debt/own revenue
F7	Operating expenses/own revenue
F8	Subsidies/population
F9	Own revenue/population
<i>Transparency variables</i>	
T1	Voter turnout
T2	Political competence
T3	Political ideology
T4	Population
T5	Unemployment
T6	Total debts
T7	Investment
T8	Political party fragmentation
T9	Political fragmentation

Variables F1, F3 and F4 refer to the structure of municipal debt. F1 shows the origin of the debt incurred by the municipality, indicating the proportion deriving from finance by suppliers (commercial debt) and the proportion arising from bank finance (financial debt). For its part, F3 represents the composition and duration of the financial debt. Variable F4 registers the dependency of a municipality on external finance. On the other hand, variables F2 and F9 show the structure of municipal revenues. The first variable measures the degree of autonomy of the municipality compared with possible subsidies received from the central government. The second variable is an indicator of municipal revenue per capita. Variables F5, F6 and F7 refer to the capacity of revenues

to pay debts and cover municipal costs. Variable F5 represents the cover provided by municipal revenues in relation to total accumulated debt. Lastly, variable F8 refers to the volume of subsidies which would correspond to each citizen.

With regard to the transparency variables, variable T1 shows the level of citizen involvement in politics and is a proxy of the demand for transparency. Variable T2 is related to the level of approval of the activities of the political parties and the level of pressure for greater transparency. Variables T4 and T5 provide information about the characteristics of the population, while variables T6 and T7 are related to investment and debt, calculated on a per capita basis. Lastly, variables T8 and T9 indicate the composition of the municipal government and the distribution of power. T8 is calculated by dividing the number of councillors belonging to the ruling party by the total number of councillors, and T9 is calculated by dividing the number of parties with representation in the local assembly by the total number of councillors.

Finally, our model incorporates as a dependent variable a new proxy of financial situation and, as mentioned above, makes reference to the ratio of default to municipal commercial debt. This is a point of reference deriving from Spanish Legislation (Organic Law 2/2012, dated 27th April 2012, regarding Budgetary Stability and Financial Solvency and Royal Decree-Law 635/2014, dated 25th July 2014) as an indicator of financial distress. According to this proposal, municipalities are considered to be in a good financial situation if the average period for paying debts is less than 30 days. For this purpose, municipalities are classified in two groups according to their average period of payment. In particular, municipalities with an average period of payment of less than 30 days are considered to be in a non-financial distress situation, while those with an average period of payment greater than 30 days are classified as being in a financial distress situation. The above-mentioned Royal Decree-Law 635/2014 stipulates the calculation of the average period of payment as the division of outstanding debts by net acknowledged debts (taking into account in both cases spending on current goods and services and real investments), multiplied by 365 days.

5 Empirical Results

The main descriptive statistics of the selected variables for this work are shown in Table 2. Municipalities in a situation of financial distress ($FD = 1$), compared with those that are not ($FD = 0$) are characterized by a higher leverage level (F1, F4), high Short-Term Debt/Own Revenue (F6), and high Operating Expenses/Own Revenue (F7). Also, they present higher average values in Total Debts (T6) and higher Political Fragmentation (T9).

Table 3 shows the results obtained with DT, NB, MLP, and DBN classifiers. The accuracy rates for the training data are 95.45%, 93.97%, 95.52%, and 94.43% respectively. With testing data (see also Fig. 2), the accuracy rates are slightly older, except for DT (79.04%, 84.35%, 93.91%, and 91.27%). Comparing the level of prediction for the model studied, a higher accuracy rate for MLP is seen. With MLP, the significant variables, in order of importance, are Short-Term Debt/Own Revenue (F6), Total Debts (T6), Operating Expenses/Own Revenue (F7), Subsidies/Population (F8), and Political Fragmentation (T9), which, taken as a whole, constitute the best set of predictors for municipal financial distress.

Table 2. Descriptive statistics

	F1	F2	F3	F4	F5	F6	F7	F8	F9	T1	T2	T3	T4	T5	T6	T7	T8	T9
Mean	0.892	2.217	1.771	0.464	0.597	0.806	1.131	78.216	1687.146	61.519	16.871	3.357	130679.156	13584.281	1230.222	5495.106	0.265	0.161
Median	0.324	1.854	1.428	0.281	0.416	0.594	0.843	45.227	1145.26	48.945	9.439	2.158	8851.373	8194.572	1027.358	4087.601	0.149	0.095
St. Dev.	2.504	3.909	2.324	0.358	2.845	1.258	0.346	33.603	825.373	7.552	13.039	1.749	1317.259	1228.028	874.170	670.118	0.107	0.044
N	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
Mean	0.410	2.324	0.613	0.315	0.667	0.450	0.953	89.885	2204.025	62.497	20.799	0.272	217184.328	19069.046	871.569	6721.257	0.096	0.145
Median	0.294	1.847	0.521	0.254	0.548	0.378	0.683	72.157	1753.548	54.324	15.279	0.168	1866536.865	16352.653	594.896	5946.974	0.049	0.114
St. Dev.	0.411	1.264	0.639	0.238	3.766	0.554	0.285	37.030	120.381	20.799	11.499	0.323	4325.542	3211.803	629.853	191.356	0.089	0.039
N	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
P-value (Kruskal-Wallis test)	0.006	0.070	0.057	0.007	0.630	0.000	0.001	0.094	0.208	0.602	0.070	0.931	0.429	0.646	0.007	0.328	0.061	0.027

Table 3. Results of accuracy evaluation

Method	Accuracy classification (%)		RMSE		Significant variables
	Training	Testing	Training	Testing	
DT	95.45	79.04	1.18	1.85	F6, F2, F5, T8, T2
NB	93.97	84.35	1.69	1.71	F1, F4, T8, T9, T1
MLP	95.52	93.91	0.97	0.92	F6, T6, F7, F8, T9
DBN	94.43	91.27	1.41	1.08	F6, T6, F5, F8, T3, T9

RMSE: Root mean squared error; Significant variables: Normalized important > 20%

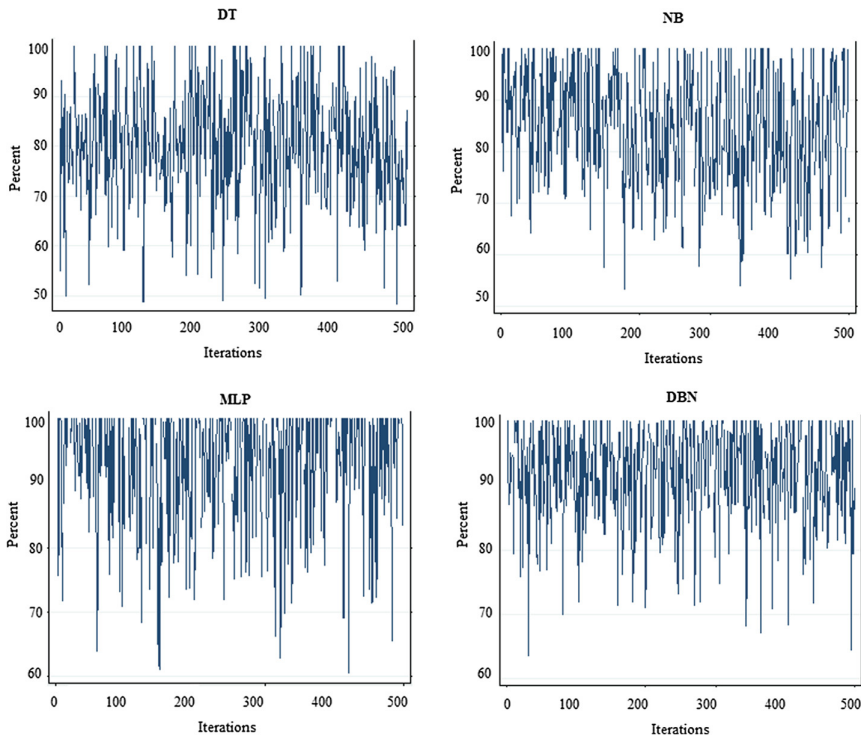


Fig. 2. Accuracy in testing data for 500 iterations

The results of this work show that the model developed increases the ability to predict municipal financial distress, compared with previous studies. Using MLP and the ratio of default to municipal commercial debt as a proxy of financial distress, the prediction accuracy is 93.91%. These results are higher than those contained in previous literature. For example, [15] using logistic regression and a multi-state dependent variable on the basis of the cause of default, obtained a level of accuracy of 76%. [3],

also using logistic regression, and with a set of financial indicators, achieved a level of accuracy of 75%. [18] used the case of municipal default as a dependent variable and their results achieved a level of accuracy of 69.14% using financial variables, and 79.73% when also including socio-economic variables.

The accuracy levels obtained exceed those achieved in previous studies, possibly also due to the larger variable set used in this study, which also includes transparency variables such as Voter Turnout and Political Ideology. Transparency is a concept which refers to the availability of information about governmental institutions and which enables citizens and other external agents to verify the performance of public institutions [29, 30]. The transparency of municipal governments is related to the financial situation thereof [9], and the use of a set of transparency variables has enhanced the predictive power of the model constructed in this work.

6 Conclusions and Implications

Municipal financial distress is a global phenomenon which has captured the attention of researchers and managers of public institutions in recent decades. In this context, numerous models have been developed for evaluating municipal financial distress and one of their main limitations is related to the measurement of financial condition, since it is not directly observed. For this reason, current literature requires new research which will permit a comparison of results using others proxies of financial situation. This work proposes a model for evaluating the financial distress of municipalities which incorporates a new proxy of their financial situation, specifically the ratio of default to municipal commercial debt. To this end, a data mining focus was applied to a sample of Spanish municipalities, and this enabled their level of financial distress to be graded convincingly using a set of variables corresponding to 2015.

Using the MLP method, our model obtained a level of accuracy greater than 93% and has successfully determined the best set variables for predicting municipal financial distress. This set includes financial variables and variables related to the transparency of municipal government. Compared with previous studies, the model developed in this work increases the ability to predict municipal financial distress, and confirms that the use of different proxies of the financial situation of a municipality provides noticeably different results.

Our results contribute to the knowledge of the financial situation of municipalities in various senses. On the one hand, it can help researchers and academics to understand how the use of certain proxies of financial situation can enhance the level of accuracy of municipal financial distress models. On the other hand, our findings could be very helpful to local government financial managers, politicians and tax authorities as we have identified factors whose evolution may influence both the viability of public services and the effectiveness of measures taken to meet the goals of budgetary stability and financial sustainability.

Future research could determine whether macroeconomic and industrial conditions such as interest rates, the age of the workforce, and industry can be good predictors for municipal financial distress prediction models.

References

1. Liao, X., Liu, Y.: Local fiscal distress and investment efficiency of local SOEs. *China J. Account. Res.* **7**(4), 119–147 (2014). <https://doi.org/10.1016/j.cjar.2013.07.002>
2. Turley, G., Robbins, G., McNena, S.: A framework to measure the financial performance of local governments. *Local Gov. Stud.* **41**(3), 401–420 (2015). <https://doi.org/10.1080/03003930.2014.991865>
3. Cohen, S., Costanzo, A., Manes-Rossi, F.: Auditors and early signals of financial distress in local governments. *Manag. Audit. J.* **32**(3), 234–250 (2017). <https://doi.org/10.1108/maj-05-2016-1371>
4. Kloha, P., Weissert, C.S., Kleine, R.: Developing and testing a composite model to predict local fiscal distress. *Public Adm. Rev.* **65**(3), 313–323 (2005). <https://doi.org/10.1111/j.1540-6210.2005.00456.x>
5. García-Sánchez, I.M., Cuadrado-Ballesteros, B., Frías-Aceituno, J.V., Mordan, N.: A new predictor of local financial distress. *Int. J. Public Adm.* **35**(11), 739–748 (2012). <https://doi.org/10.5539/ijbm.v7n1p169>
6. Honadle, B.W., Costa, J.M., Cliger, B.A.: *Fiscal Health for Local Governments: An Introduction to Concepts, Practical Analysis, and Strategies*. Elsevier Academic Press, San Diego (2004). <https://doi.org/10.1016/b978-012354751-4.50010-x>
7. Cuadrado-Ballesteros, B., Mordán, N., García-Sánchez, I.M.: Is local financial health associated with citizens' quality of life? *Soc. Ind. Res.* **119**, 559–580 (2014). <https://doi.org/10.1007/s11205-013-0533-2>
8. Zafra-Gómez, J.L., López-Hernández, A.M., Hernández-Bastida, A.: Developing a model to measure financial condition in local government. *Am. Rev. Public Adm.* **39**(4), 425–449 (2009). <https://doi.org/10.22146/jieb.v29i2.6206>
9. Ferreira, A.C.S., Do Carmo Azevedo, G.M., Da Silva Oliveira, J., Marques, R.P.F.: *Global Perspectives on Risk Management and Accounting in the Public Sector*. Elsevier B.V. (2016). <https://doi.org/10.4018/978-1-4666-9803-1>
10. Honadle, B.W.: The states' role in US local government fiscal crises: a theoretical model and results of a national survey. *Int. J. Public Adm.* **26**(13), 1431–1472 (2003). <https://doi.org/10.1081/pad-120024405>
11. Jones, S., Walker, R.: Explanators of local government distress. *Abacus* **43**(3), 396–418 (2007). <https://doi.org/10.1111/j.1467-6281.2007.00238.x>
12. Gorina, E., Maher, C., Joffe, M.: Local fiscal distress: measurement and prediction. *Public Budgeting Finan.* (2017). <https://doi.org/10.1111/pbaf.12165>
13. Pina, V., Torres, L., Yetano, A.: Accrual accounting in EU local governments: one method, several approaches. *Eur. Account. Rev.* **18**(4), 765–807 (2009). <https://doi.org/10.1080/09638180903118694>
14. Cohen, S., Doumpos, M., Neofytou, E., Zopounidis, C.: Assessing financial distress where bankruptcy is not an option: an alternative approach for local municipalities. *Eur. J. Oper. Res.* **218**, 270–279 (2012). <https://doi.org/10.1016/j.ejor.2011.10.021>
15. Navarro-Galera, A., Rayo-Cantón, S., Lara-Rubio, J., Buendía-Carrillo, D.: Loan price modelling for local governments using risk premium analysis. *Appl. Econ.* **47**(58), 6257–6276 (2015). <https://doi.org/10.1080/00036846.2015.1068924>
16. Navarro-Galera, A., Rodríguez-Bolívar, M.P., Alcaide-Muñoz, L., López-Subires, M.D.: Measuring the financial sustainability and its influential factors in local governments. *Appl. Econ.* **48**(41), 3961–3975 (2016). <https://doi.org/10.1080/00036846.2016.1148260>

17. Galariotis, E., Guyot, A., Doumpou, M., Zopounidis, C.: A novel multi-attribute benchmarking approach for assessing the financial performance of local governments: empirical evidence from France. *Eur. J. Oper. Res.* **248**, 301–317 (2016). <https://doi.org/10.1016/j.ejor.2015.06.042>
18. Lara-Rubio, J., Rayo-Cantón, S., Navarro-Galera, A., Buendía-Carrillo, D.: Analysing credit risk in large local governments: an empirical study in Spain. *Local Gov. Stud.* **43**(2), 194–217 (2017). <https://doi.org/10.1080/03003930.2016.1261700>
19. Sun, J., Li, H.: Data mining method for listed companies' financial distress prediction. *Knowl.-Based Syst.* **21**, 1–5 (2008). <https://doi.org/10.1016/j.knosys.2006.11.003>
20. Callejón, A.M., Casado, A.M., Fernández, M.A., Peláez, J.I.: A system of insolvency prediction for industrial companies using a financial alternative model with neural networks. *Int. J. Comput. Intell. Syst.* **6**(1), 29–37 (2013). <https://doi.org/10.1080/18756891.2013.754167>
21. Kingsford, C., Salzberg, S.L.: What are decision trees? *Nat. Biotechnol.* **26**, 1011–1013 (2008). <https://doi.org/10.1038/nbt0908-1011>
22. Escalante, H.J., Morales, E.F., Sucar, L.E.: A naïve Bayes baseline for early gesture recognition. *Pattern Recogn. Lett.* **73**, 91–99 (2016). <https://doi.org/10.1016/j.patrec.2016.01.013>
23. Feki-Sahnoun, W., Njah, H., Hamza, A., Barraji, N., Mahfoudi, M., Rebai, A., Hassen, M.B.: Using general linear model, Bayesian networks and Naive Bayes classifier for prediction of *Karenia selliformis* occurrences and blooms. *Ecol. Inf.* **43**, 12–23 (2018). <https://doi.org/10.1016/j.ecoinf.2017.10.017>
24. Lewis, D.D.: Naive (Bayes) at forty: the independence assumption in information retrieval. In: *European Conference on Machine Learning*, pp. 4–15 (1998). <https://doi.org/10.1007/bfb0026666>
25. De Castro, L.N., Iyoda, E.M., Von Zuben, F.J., Gudwin, R.: Feedforward neural network initialization: an evolutionary approach. In: *Proceedings 5th Brazilian Symposium on Neural Networks*. Belo Horizonte, Brazil, pp. 43–48 (1998). <https://doi.org/10.1109/sbrn.1998.730992>
26. Bengio, Y.: Learning deep architectures for artificial intelligence. *Found. Trends Mach. Learn.* **2**(1), 1–127 (2009). <https://doi.org/10.1561/22000000006>
27. Flórez, R., Fernández, J.M.: *Las Redes Neuronales Artificiales. Fundamentos teóricos y aplicaciones prácticas*. Ed. Netbiblo. Coruña (2008). <https://doi.org/10.4272/978-84-9745-246-5.ch1>
28. Araujo, J.F., Tejedo-Romero, F.: Local government transparency index: determinants of municipalities' rankings. *Int. J. Public Sect. Manag.* **29**(4), 327–347 (2016). <https://doi.org/10.1108/ijpsm-11-2015-0199>
29. Meijer, A.: Understanding the complex dynamics of transparency. *Public Adm. Rev.* **73**(3), 429–439 (2013). <https://doi.org/10.1111/puar.12032>
30. Grimmelikhuijsen, S.: Linking transparency, knowledge and citizen trust in government: an experiment. *Int. Rev. Adm. Sci.* **78**(1), 50–73 (2012). <https://doi.org/10.1177/0020852311429667>



Prefix and Suffix Sequential Pattern Mining

Rina Singh^(✉), Jeffrey A. Graves, Douglas A. Talbert, and William Eberle

Department of Computer Science, Tennessee Technological University,
Cookeville, USA
rsingh43@students.tntech.edu

Abstract. Sequential pattern mining is a challenging problem that has received much attention in the past few decades. The mining of large sequential databases can be very time consuming and produces a large number of unrelated patterns that must be evaluated. In this paper, we explore the problems of frequent prefix, prefix-closed, and prefix-maximal pattern mining along with their suffix variants. By constraining the pattern mining task, we are able to reduce the mining time required while obtaining patterns of interest. We introduce notations related to prefix/suffix sequential pattern mining while providing theorems and proofs that are key to our proposed algorithms. We show that the use of projected databases can greatly reduce the time required to mine the complete set of frequent prefix/suffix patterns, prefix/suffix-closed patterns, and prefix/suffix-maximal patterns. Theoretical analysis shows that our approach is better than the current existing approach, and empirical analysis on various datasets is used to support these conclusions.

1 Introduction

For the past several decades, ever since Agrawal and Srikant first published their paper, Mining Sequential Patterns [1], sequential pattern mining has been of broad and current interest. Sequential pattern mining was originally explored for mining information from customer transaction databases. It has been used to predict purchasing behavior [2], applied to next-item prediction problems in recommender systems [3], used to help define guidelines for patient care [4], informed the verification and development of clinical pathways [5], and even guided product placement within supermarkets [6]. Sequential pattern mining has also been used to suggest customer relationship management strategies for small online businesses [7]. However, as transaction databases continue to grow in size and scope, the time required to mine useful patterns continues to increase. Furthermore, actionable retail information is often time sensitive, and while it is possible to obtain patterns in a reasonable (i.e., short) amount of time using many sequential pattern mining techniques, the patterns obtained are often too small or too general to be useful.

While the basic sequential pattern mining task is unconstrained, several constrained variants have been developed. Some of these constraint-based

approaches can help to reduce mining time while allowing a more narrow focus on the patterns obtained. Examples include mining patterns with recency and compactness constraints [8], gap constraints [9–11], relaxation of itemset/transaction constraints [11], and taxonomy based constraints [11].

Despite these constraint variations of sequential pattern mining, most solutions are ill-suited for identifying all sequences that begin or end with an event or events of interest. We are particularly interested in this task, because we are seeking to help improve the selection of cancer treatments through data-driven modeling and simulation of health trajectories. We see frequent sequence mining as a tool that can help with this. In particular, we see the mining of frequent health data sequences that begin or end with specific cancer treatments of interest as potentially informative. Knowing such sequences could allow us to discover probabilistic rules that link sequences that precede particular cancer treatments to subsequent health trajectories.

To advance us toward our goal, this paper focuses on solving this version of constraint-based sequential pattern mining. We provide formal definitions and notation related to mining patterns having a user-defined prefix or suffix. Several theorems are formulated and proven, which provide insight for developing efficient mining techniques for these prefix and suffix patterns. Our newly proposed algorithms are described, and their weaknesses and strengths are compared to the only previously published existing approach for prefix/suffix pattern mining. We have provided theoretical and empirical analysis demonstrating the improvement of our proposed algorithms over the approach proposed by Kaytoue et al. in their work on mining graph sequences [12]. The extraction of cancer-related data is underway but not yet complete, so empirical analysis has been performed on several other real-world datasets in various domains (e.g., click-streams, retail sales, etc.).

2 Sequential Pattern Mining

Closely related to sequential pattern mining is frequent itemset mining. While frequent itemset mining focuses on discovering statistically relevant co-occurring items, sequential pattern mining focuses on identifying patterns among itemsets presented as a sequence. Great effort has been put forth by the data mining community to mine sequential patterns. A number of algorithms have been proposed to mine the complete collection of frequent patterns (i.e., sequences) from a sequential database. Examples include SPADE [13], SPAM [14], and PrefixSpan [15]. Sequential pattern mining can produce large result sets, partially due to the combinatorial nature of the mining task and redundant information represented in the results. To eliminate this redundant information, several algorithms have been proposed to mine frequent closed patterns and frequent maximal patterns. BIDE [16] and CloSpan [17] are examples of such algorithms.

However, as pointed out by Kaytoue et al. none of these algorithms can directly mine patterns given a user-defined prefix or suffix and must be adapted [12]. Prefix and suffix patterns can be useful for uncovering cause and

effect relationships. For instance, we are interested in identifying the most common health trajectories occurring after a particular cancer treatment. In this case, the cancer treatment serves as a prefix, for which we want to know the effects.

Fournier–Viger et al. provide an excellent survey on sequential pattern mining [18], and it will be assumed that the reader is familiar with basic definitions, notation, and results from sequential pattern mining. The following section introduces definitions, notation, and results related to prefix/suffix pattern mining used throughout this paper.

3 Prefix/Suffix Pattern Mining

In their research, Kaytoue et al. focus on describing topological changes in dynamic attributed graphs [12]. In order to accomplish this, the authors formally defined the notion of a prefix-closed pattern and a variant on the sequential pattern mining problem they call prefix-closed pattern mining.

3.1 Prefix/Suffix Sequences

Given a fixed sequence A , one may be interested in sequences that “begin” with A or “end” with A . In the former case, A serves as a prefix, and the patterns of interest are called suffix patterns. In the latter case, A serves as a suffix, and the patterns of interest are called prefix patterns. Note that no restrictions are placed on A , and it can be of arbitrary length and could be a sequence of items (i.e., an item sequence) or a sequence of sets of items (i.e., an itemset sequence).

Definition 1. Let $A = \{a_i\}_{i=1}^m$ and $B = \{b_i\}_{i=1}^n$ be sequences. Let $C = A \oplus B$ denote the sequence $\{c_i\}_{i=1}^{m+n}$ given by **juxtaposition** of A and B , where

$$c_i = \begin{cases} a_i & 1 \leq i \leq m \\ b_{m-i} & m < i \leq m+n. \end{cases}$$

Definition 2. Let P and S be non-empty sequences and let $A = P \oplus S$. The sequence P is called a **prefix** of A and the sequence S is called a **suffix** of A . The sequence A is sometimes called a **P -prefix sequence** or an **S -suffix sequence**. Alternatively, let $A = \{a_i\}_{i=1}^m$ and $B = \{b_i\}_{i=1}^n$ be sequences with $m < n$. If $a_i = b_i$ for $1 \leq i \leq m$, then A is called a **prefix** of B . If $a_i = b_{n-m+i}$ for all $1 \leq i \leq m$, then A is called a **suffix** of B .

Note that, unlike in the definition of a subsequence, we require equality of elements (both items and itemsets) in the definition of a prefix/suffix. For example, the sequence $\langle \{a\}, \{b\} \rangle$ is a prefix of $\langle \{a\}, \{b\}, \{c\} \rangle$, but it is not a prefix of $\langle \{a, b\}, \{c\} \rangle$, $\langle \{a, x\}, \{b, y\}, \{c\} \rangle$, or $\langle \{a\}, \{x\}, \{b\}, \{y\} \rangle$.

Definition 3. A sequence A is said to be **prefix-closed**, with respect to a database D and suffix B , if B is a suffix of A (that is to say, $A = A' \oplus B$ for some non-empty sequence A') and no proper supersequence of A , also having suffix B , has the same support as A in D . A sequence A is said to be **prefix-maximal**, with respect to a set S and suffix B , if B is a suffix of A and there exists no proper supersequence of A in S also having suffix B . Analogous definitions exist for **suffix-closed** and **suffix-maximal** sequences as well.

Consider the example sequential database presented Fig. 1. The sequence $\langle \{b\}, \{f\}, \{e\} \rangle$ is a $\langle \{f\}, \{e\} \rangle$ -suffix sequence. It is prefix-closed, and has a support of two, but it is not prefix-maximal in the collection of all frequent prefix sequences with positive support as it is a subsequence of $\langle \{a, b\}, \{f\}, \{e\} \rangle$, which is also a $\langle \{f\}, \{e\} \rangle$ -suffix sequence of positive support.

Again, consider the sequential database from Fig. 1. The sequence $\langle \{f, g\}, \{g\} \rangle$ is a $\langle \{f, g\} \rangle$ -prefix sequence with a support of one. It is not suffix-closed as it is a subsequence of $\langle \{f, g\}, \{g\}, \{e\} \rangle$, which is also $\langle \{f, g\} \rangle$ -prefix sequence with the same support; the sequence $\langle \{f, g\}, \{g\}, \{e\} \rangle$ is both suffix-closed and suffix-maximal (in the set of frequent suffix patterns with positive support).

ID	Sequence
1	$\langle \{a, b\}, \{c\}, \{f, g\}, \{g\}, \{e\} \rangle$
2	$\langle \{a, d\}, \{c\}, \{b\}, \{a, b, e, f\} \rangle$
3	$\langle \{a\}, \{b\}, \{f\}, \{e\} \rangle$
4	$\langle \{b\}, \{f, g\} \rangle$

Fig. 1. A sample sequential database

It is worth noting that, just like with closed/maximal sequences, every prefix-maximal sequence is prefix-closed, but not every prefix-closed sequence is prefix-maximal, and same thing is true for suffix-closed and suffix-maximal [15, 17].

3.2 Prefix/Suffix Projections

Database projections are often used to help reduce the number of database scans and eliminate redundant computation during the mining task [15, 17]. In our proposed algorithms, we make use of database projections as a preprocessing step in order to reduce the search space associated with prefix/suffix mining task. This section introduces definitions and notation involving prefix and suffix projections related to our proposed algorithms.

Definition 4. Let A and B be sequences. The **prefix-projection** of A by B (or **B -prefix-projection** of A) is the longest subsequence A' of A such that $A = B' \oplus A'$ and $B \preceq B'$, which is denoted $\mathcal{P}_B(A) = A'$. If no such subsequence exists, the prefix-projection is defined to be the empty sequence.

Definition 5. Let A and B be sequences. The **suffix-projection** of A by B (or **B -suffix-projection** of A) is the longest subsequence A' of A such that $A = A' \oplus B'$ and $B \preceq B'$, which is denoted $\mathcal{S}_B(A) = A'$. If no such subsequence exists, the suffix-projection is defined to be the empty sequence.

Definition 6. Let S be a sequence and D be a sequential database. The collection of all S -prefix-projected sequences in D , denoted $\mathcal{P}_S^*(D)$, is called the **S -prefix-projected database** of D . The collection of all S -suffix-projected sequences in D , denoted $\mathcal{S}_S^*(D)$, is called the **S -suffix-projected database** of D .

3.3 Sequential Prefix/Suffix Mining Problems

While the initial goal of Kaytoue et al. was not to advance the area of sequential pattern mining problem, they were the first to propose the prefix-closed mining variant of the sequential pattern mining problem.

Problem 1. Let D be a sequential database, S be a sequence, and n a user-defined minimum support. The **prefix (closed/maximal) sequential pattern mining problem** asks to find the complete set of frequent prefix (closed/maximal) patterns with respect to database D and suffix S . The **suffix (closed/maximal) sequential pattern mining problem** asks to find the complete set of frequent suffix (closed/maximal) patterns with respect to database D and prefix S .

Kaytoue and his fellow authors claim that the adaption of closed patterns to prefix-closed patterns is not straightforward and proposed a new algorithm. This algorithm is based on a new theorem, developed and proven by Kaytoue and his colleagues, which states that the collection of frequent prefix-closed patterns are contained within the collection of closed patterns.

Theorem 1. For every frequent prefix-closed pattern A , there exists a frequent closed pattern B such that $A \preceq B$ and $\text{support}(A) = \text{support}(B)$.

In their work on dynamic attributed graphs, Kaytoue and his colleagues only mentioned prefix-closed patterns. However, their observations hold for prefix-maximal patterns and analogous results can be formulated for suffix-closed and suffix-maximal patterns. While this result is interesting from a theoretic viewpoint, we claim that it does not provide a basis for the most efficient algorithm for solving the prefix-closed mining problem. We have proven the following theorem, which provides an alternate method for finding the set of frequent prefix, prefix-closed, and prefix-maximal patterns.

Theorem 2. Let A be a sequence, n be a user-defined minimum support, and D be a sequential database. Then,

- (i) The complete collection of frequent prefix patterns with respect to database D having suffix A and minimum support n is given by $\{P = P' \oplus A \mid P' \text{ is frequent in } \mathcal{S}_A^*(D)\}$.

- (ii) The complete collection of prefix-closed patterns with respect to database D having suffix A and minimum support n is given by $\{P = P' \oplus A \mid P' \text{ is closed in } \mathcal{S}_A^*(D)\}$.
- (iii) The complete collection of prefix-maximal patterns with respect to database D having suffix A and minimum support n is given by $\{P = P' \oplus A \mid P' \text{ is maximal in } \mathcal{S}_A^*(D)\}$.

Analogous results for the suffix variation of Theorem 2 exists but have been omitted. Part (i) in Theorem 2 is not surprising and can be established with a fairly straightforward direct proof. What is not so obvious are Parts (ii) and (iii). Let A be a sequence, n a minimum support, and D a sequential database. If P' is a closed/maximal pattern in $\mathcal{S}_A^*(D)$, why must the prefix pattern $P = P' \oplus A$ also be prefix-closed/maximal in D ? If $P = P' \oplus A$ is prefix-closed/maximal pattern in D , why must P' also be closed/maximal in $\mathcal{S}_A^*(D)$?

Proof. The proof of Part (iii) is analogous to that of Part (ii). To establish (ii), we need only to show that every prefix-closed pattern having suffix A is of the form $P = P' \oplus A$ for some sequence P' which is closed in $\mathcal{S}_A^*(D)$, and that every sequence of the form $P = P' \oplus A$ where P' is closed in $\mathcal{S}_A^*(D)$ is a prefix-closed pattern in D .

(\subseteq) Suppose that P is a prefix-closed pattern with respect to suffix A in database D . Then $P = P' \oplus A$ for some non-empty sequence P' . We proceed by way of contradiction. Suppose that P' is not closed in $\mathcal{S}_A^*(D)$. Then, there exists a supersequence, say P'' , such that $P' \prec P''$ and $support_{\mathcal{S}_A^*(D)}(P') = support_{\mathcal{S}_A^*(D)}(P'')$. Put $S = P'' \oplus A$, and observe that $P \prec S$. As established in the proof of Theorem 2, we have that $support_D(P) = support_{\mathcal{S}_A^*(D)}(P')$ and $support_D(S) = support_{\mathcal{S}_A^*(D)}(P'')$. It follows that $support_D(P) = support_D(S)$, a contradiction to the assumption that P was closed.

(\supseteq) Now, suppose that $P = P' \oplus A$ for some closed sequence P' in $\mathcal{S}_A^*(D)$. We want to show that P is prefix-closed. Suppose not. That is, suppose that there exists an A -suffix sequence, say S , such that $P \prec S$ and $support_D(P) = support_D(S)$. Since S and $P = P' \oplus A$ are both A -suffix sequences, we have that $S = P'' \oplus A$ for some sequence $P'' \prec P'$. Since $support_D(P) = support_{\mathcal{S}_A^*(D)}(P')$, $support_D(S) = support_{\mathcal{S}_A^*(D)}(P'')$, and $support_D(P) = support_D(S)$, it is the case that $support_{\mathcal{S}_A^*(D)}(P') = support_{\mathcal{S}_A^*(D)}(P'')$. That is to say, P' is not closed, a contradiction.

3.4 Algorithms

Upon first thought, one may think that there should be a relationship between the complete set of frequent prefix patterns, prefix-closed patterns, and prefix-maximal patterns and the complete set of frequent patterns, closed patterns, and maximal patterns, respectively. Theorem 1 provided by Kaytoue, et al., proves this to be true [12]. To mine the complete set of frequent prefix-closed patterns with respect to a database D and a suffix S , begin by mining the complete set of frequent closed patterns in D . Then for each pattern A , consider the S -suffix-projected sequence $\mathcal{S}_S(A) = A'$ of A . If A' is non-empty, the sequence $A' \oplus S$

is a potential prefix-closed sequences. Some of the sequences obtained from this suffix projection-extension process may not be prefix-closed. After filtering out the non-prefix-closed patterns, the complete set of frequent prefix-closed patterns will be all that remains. Kaytoue et al. use a subsumption checking algorithm to perform this filtering. This idea is summarized in Algorithm 1; analogous algorithms exist for prefix pattern mining, prefix-maximal pattern mining, and their suffix mining variants.

Algorithm 1. Kaytoue Based Prefix-Closed Pattern Mining

Require: D : a sequential database
Require: S : a suffix for prefix mining
Require: n : a user-defined minimum support

```

1: procedure CLOSEPREFIXMINER( $D, S, n$ )
2:    $C \leftarrow$  CLOSEDPATTERNMINING( $D, n$ )
3:    $C' \leftarrow []$ 
4:   for  $A' \in C$  do
5:     if  $\mathcal{S}_S(A') \neq \langle \rangle$  then
6:        $A \leftarrow \mathcal{S}_S(A') \oplus S$ 
7:        $C' \leftarrow C' + [A]$ 
8:     end if
9:   end for
10:   $C' \leftarrow$  SUBSUMPTIONFILTERING( $C'$ )
11:  return  $C'$ 
12: end procedure

```

Theorem 2 provides an alternate approach for obtaining the complete set of frequent prefix, prefix-closed, and prefix-maximal patterns with an analogous version existing for their suffix counterparts. It is fairly easy to see that the complete set of frequent prefix patterns can be obtained from a suffix-projected database. What is not so obvious is that the complete set of prefix-closed and prefix-maximal patterns (along with their suffix variants) corresponds directly to the complete set of prefix-closed and prefix-maximal patterns found in the associated suffix-projected database. For example, to obtain the complete set of prefix-closed patterns from a database D having suffix S , begin by constructing the suffix-projected database $\mathcal{S}_S^*(D)$. Then, mine the complete set of closed patterns from $\mathcal{S}_S^*(D)$. For each closed pattern P obtained, the sequence $P \oplus S$ is guaranteed to be prefix-closed, by Theorem 2. This idea is summarized in Algorithm 2; analogous algorithms exist for prefix pattern mining, prefix-maximal pattern mining, and their suffix mining variants. In Sect. 4, we will demonstrate the advantage of Algorithm 2 over Algorithm 1.

If one is interested in obtaining prefix-closed patterns with respect to multiple suffixes, these algorithms can be modified in order to obtain the desired results. In the case of Algorithm 1, we need only to mine the complete set of closed patterns once. Then, each of the suffixes can be used in turn to extract the complete set of prefix-closed patterns, as shown in Algorithm 3. The extension to Algorithm 2 is a little more complicated. For each of the desired suffixes, a suffix-projected database must first be constructed. Then the complete set of closed patterns can be mined (see Algorithm 4). Having to mine multiple projected

databases for closed patterns may seem inefficient, but as we will demonstrate in Sect. 4, this is actually faster.

Algorithm 2. Efficient Prefix-Closed Pattern Mining

Require: D : a sequential database
Require: S : a suffix for prefix mining
Require: n : a user-defined minimum support

- 1: **procedure** CLOSEPREFIXMINER(D, S, n)
- 2: $D' \leftarrow \mathcal{S}_S^*(D)$
- 3: $C \leftarrow \text{CLOSEDPATTERNMINING}(D', n)$
- 4: $C' \leftarrow []$
- 5: **for** $A' \in C$ **do**
- 6: $A \leftarrow A' \oplus S$
- 7: $C' \leftarrow C' + [A]$
- 8: **end for**
- 9: **return** C'
- 10: **end procedure**

Algorithm 3. Kaytoue Based Prefix-Closed Pattern Mining w/ Multiple Suffixes

Require: D : a sequential database
Require: L_S : a list of suffixes for prefix mining
Require: n : a user-defined minimum support

- 1: **procedure** CLOSEPREFIXMINER++(D, L_S, n)
- 2: $C \leftarrow \text{CLOSEDPATTERNMINING}(D, n)$
- 3: $C' \leftarrow []$
- 4: **for** $S \in L_S$ **do**
- 5: $C'_S \leftarrow []$
- 6: **for** $A' \in C$ **do**
- 7: **if** $\mathcal{S}_S(A') \neq \langle \rangle$ **then**
- 8: $A \leftarrow \mathcal{S}_S(A') \oplus S$
- 9: $C'_S \leftarrow C'_S + [A]$
- 10: **end if**
- 11: **end for**
- 12: $C'_S \leftarrow \text{SUBSUMPTIONFILTERING}(C'_S)$
- 13: $C' \leftarrow C' + C'_S$
- 14: **end for**
- 15: **return** C'
- 16: **end procedure**

Algorithm 4. Efficient Prefix-Closed Pattern Mining w/ Multiple Suffixes

Require: D : a sequential database
Require: L_S : a list of suffixes for prefix mining
Require: n : a user-defined minimum support

- 1: **procedure** CLOSEPREFIXMINER++(D, L_S, n)
- 2: $C' \leftarrow []$
- 3: **for** $S \in L_S$ **do**
- 4: $C'_S \leftarrow \text{CLOSEPREFIXMINER}(D, S, n)$
- 5: $C' \leftarrow C' + C'_S$
- 6: **end for**
- 7: **return** C'
- 8: **end procedure**

4 Theoretical Analysis

In this section, we will illustrate the advantages of mining projected databases (i.e., Algorithms 2 and 4) over that of the full sequential database (i.e., Algorithms 1 and 3). We should first point out, that while our approach is in the same computational complexity class as Kaytoue’s approach, it can result in drastically shorter runtime. This can be attributed to a reduced search space or a reduced input problem size, depending on the point of view.

4.1 Reduced Problem Size

It is obvious that Algorithm 2 has a smaller database to mine than Algorithm 1; the smaller the projected-database is compared to the original database, the faster the closed-pattern mining task will be. First, note that the creation of the projected database is linear in terms of the number of sequences in the database, the length of the sequences in the database, and the length of the suffix in question. On the other hand, the mining task is exponential in terms of the length of the sequences being mined. To see this, let \mathcal{I} be a collection of items. The number of non-empty item sequences of length at most k , for some positive constant k , is given by

$$\sum_{i=1}^k |\mathcal{I}|^i = \frac{|\mathcal{I}|^{k+1} - |\mathcal{I}|}{|\mathcal{I}| - 1}.$$

In the case of itemset sequences, the number of non-empty subsets of \mathcal{I} is $2^{|\mathcal{I}|-1}$, and so the number of itemset sequences of length at most k is given by

$$\sum_{i=1}^k (2^{|\mathcal{I}|-1})^i = \frac{(2^{|\mathcal{I}|-1})^{k+1} - (2^{|\mathcal{I}|-1})}{(2^{|\mathcal{I}|-1}) - 1}.$$

Hence, for a fixed itemset \mathcal{I} , the number of sequences of length at most k grows exponentially with k .

In the case of sequential pattern mining, the value of k is determined by the length of the sequences in the database. More specifically, if the number of sequences in the database of length at least k is fewer than a user-defined minimum absolute support m , then no sequence of length k will be frequent. Let f_D denote the function that counts the number of sequences in a database D of length at least n , which is given by $f_D(n) = |\{S \in D \mid |S| \geq n\}|$. Then $\max\{n \mid f_D(n) \geq m\}$, where m is a user-defined minimum absolute support, places an upper bound on the value of k . And so, Algorithm 2 performs a linear number of computations in order to reduce the input size of the exponential closed pattern mining task.

The length of the suffix and its support within the original database will affect the size of the projected database. The smallest reduction in database size occurs when the suffix used in prefix-closed mining is a suffix of every sequence

in the database. In this case, every one of the sequences in the suffix-projected database is smaller than their corresponding sequence in the original database by precisely the size of the suffix used to obtain the projections. We expect the smallest possible speedup of Algorithm 2 over Algorithm 1 to occur when the suffix of interest is of length one. In practice, Algorithm 2 may be slower due to the overhead of constructing the sequential database.

Conversely, the largest reduction in database size will occur when the the suffix in question has low support in the given database. In this situation, many suffix-projected sequences within the suffix-projected database will be empty. In the extreme, the suffix will have zero support within the database, and the closed pattern mining step in Algorithm 2 will return almost immediately. This will not be the case for Algorithm 1, which will proceed to mine the complete set of closed patterns, none of which will contain the suffix of interest. In this case, the largest speedup should occur.

4.2 Reduced Search Space

An alternative explanation for the improved running time of Algorithm 2 over Algorithm 1 is the smaller search space explored by the former algorithm. Figure 2 depicts the standard search tree used to generate all item sequences given an itemset $\mathcal{I} = \{a, b, c\}$; a similar tree exists for generating all itemset sequences. Many frequent sequential pattern mining algorithms implicitly search this tree of all sequential patterns using various techniques for pruning. Consider instead the frequent suffix-closed mining variant of Algorithm 2. This algorithm will produce the complete set of frequent suffix-closed patterns, given a database D , prefix P , and minimum support n . By first constructing a P -prefix projected database, this new algorithm can be thought of as exploring a subspace of the complete search tree seen in Fig. 2. For example, given a prefix of $\langle b, a \rangle$, only the ba subtree shaded in Fig. 2 needs to be explored.

4.3 Mining Several Prefix/Suffix Patterns

On the surface, Algorithm 4 may appear worse than Algorithm 3 for mining prefix-closed patterns when multiple suffixes of interest are given. Obviously it

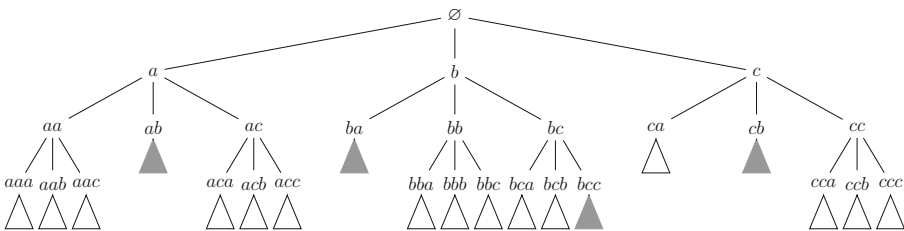


Fig. 2. Standard sequence space with $\mathcal{I} = \{a, b, c\}$

must solve several instances of the closed pattern mining problem while Algorithm 3 only has to solve one instance. Again, we will explore the dual frequent suffix-closed pattern mining algorithm to see that the use of projected sequences is still beneficial.

Given several prefixes of interest, the suffix-closed variant of Algorithm 4 can be thought to explore several subtrees within the complete sequence search tree. So long as none of the given prefixes happens to be a prefix of another, these subtrees are disjoint within the search space of all sequences. For example, given an $\mathcal{I} = \{a, b, c\}$ with suffixes $\langle a, b \rangle$, $\langle b, a \rangle$, $\langle c, b \rangle$, and $\langle b, c, c \rangle$ of interest, only the ab , ba , cb , bcc subtrees shaded in Fig. 2 need to be explored. As the number of prefixes for use in suffix-closed mining increase, larger portions of the search tree must be explored; if every possible item is included as a prefix sequence of length one, then the entire space must be explored. In this case, Algorithm 4 should not be expected to provide any speedup over Algorithm 3, and may be slower due to the overhead of building the projected databases.

5 Empirical Analysis

In order to show that our proposed algorithms are useful in practice, we have implemented Algorithms 1 to 4 using C++. It is worth noting that, for ease of implementation, physical prefix-projected and suffix-projected databases were created for use by the sequential pattern miner in Algorithms 2 and 4 (pseudo-projections were used within the actual sequential pattern miners). This allows for easy replacement of the frequent/closed/maximal mining algorithm based on dataset characteristics. This is desirable as some algorithms work better on long sequences with few possible items while others work better on short sequences with many possible items [18].

However, the use of physical projected databases will result in a larger memory footprint and additional overhead when creating the physical projection. The use of a pseudo-projected database would eliminate much of the overhead associated with creating the prefix-projected and suffix-projected databases. In addition, it would require no more memory than that of Algorithms 1 and 3 because of the smaller search space examined. The downside is that some sequential pattern minings would require modification if a pseudo-projection is used. For example, the *backward-extension checking* step used in the BIDE closed-pattern mining algorithm [16] would eliminate a suffix-closed pattern, with prefix P , of the form $P \oplus A$ if there exists a supersequence $P' \oplus A$, where $P \prec P'$, having the same support; this is a problem since P' would not have prefix P .

5.1 Empirical Setup

We have tested our algorithm on several sequential databases from various domains, with datasets taken primarily from Fournier-Viger's SPMF website [19]. While there are several constraint-based sequential pattern mining problems, Kaytoue's proposed approach is the only one that attempts to solve

the prefix/suffixed closed pattern mining problem, and so we use it as a baseline for comparison (i.e., Algorithms 1 and 3 against Algorithms 2 and 4). Experiments were performed on DELL c6320 servers. Each machine contained dual Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40 GHz (28 physical cores) and 128 GB of RAM; the algorithms implemented were serial and could not take advantage of multiple cores. We chose to focus on the prefix mining variant simply because these are of most interest to us. The suffix based variants produced similar results and have been omitted for brevity. In addition, the results presented will focus on prefix-patterns where the suffix of interest is a sequence consisting of a single item. This decision was made as it addresses the most difficult prefix-closed mining task; mining sequences of itemsets or allowing for suffixes of length greater than one affords more opportunities for search space pruning, and we wish to focus on worst-case situations for our proposed algorithms.

5.2 Retail

The Retail dataset consists of transactions from a UK-based and registered non-store online retail [20]. Timestamp information along with invoice ids and customer ids were used to build sequences, resulting in 4,372 sequences with 3,684 items. Each sequence represents a customer, while the items represent unique all-occasion gifts sold by the retail. Since multiple items can appear on an invoice, this sequential database consists of itemset sequences, with the largest itemset consisting of 541 items. Figure 3 illustrates the time required to mine all prefix-closed patterns using the top five most and least supported suffixes. These plots show that, when the mining task is difficult (i.e., the minimum support is low or the support of the suffix is low), Algorithm 4 significantly outperforms Algorithm 3. This is due to the fact that the mining time greatly exceeds the time required to build the projected databases. Conversely, if the mining task is easy, Algorithm 4 may perform worse than Algorithm 3 due to the overhead associated with building the projected databases.

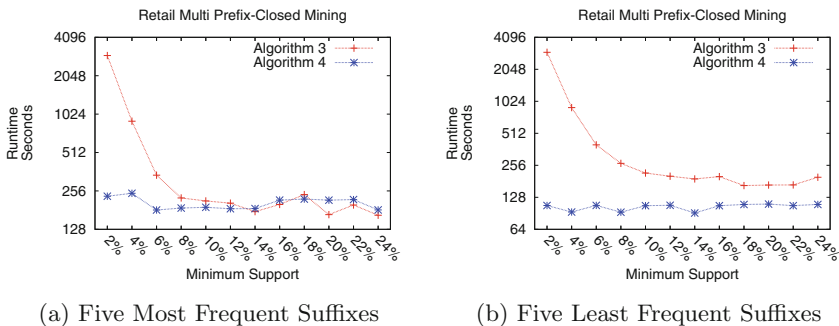


Fig. 3. Retail dataset

5.3 Click-Streams

The Gazelle dataset (i.e., the KDD CUP 2000 dataset) contains click-stream data relating to purchases from a web retailer. It contains of 77,512 sequences consisting of 3,340 unique items, with the most frequent item having 4.86% support. We selected the most and least frequently occurring items to serve as suffixes to test Algorithms 1 and 2. The runtime results can be seen in Fig. 4, which shows that Algorithm 2 outperforms Algorithm 1.

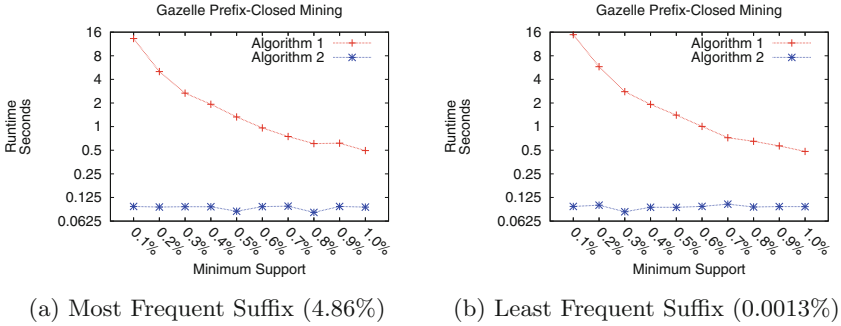


Fig. 4. Gazelle dataset

The FIFA dataset consists of 20,450 sequences made up of 2,990 items representing click-streams gathered from the FIFA World Cup 98 website. The top ten most frequently viewed webpages all had over 35% support within the database. There were several pages that were only viewed a single time resulting in a support less than 0.005%. Figure 5 depicts the runtimes for mining the single item suffix with the highest and lowest support. The most frequently occurring item within the FIFA dataset poses more difficulty for mining, but Algorithm 2 still outperforms Algorithm 1.

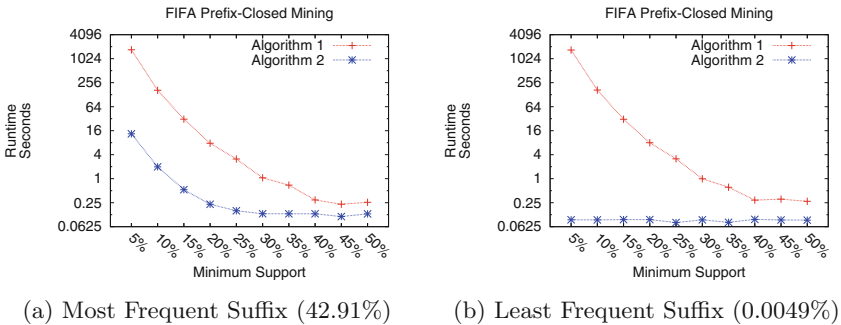
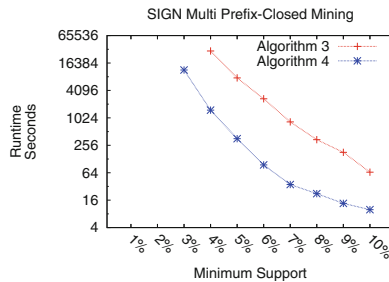


Fig. 5. FIFA dataset

5.4 Natural Languages

The SIGN dataset consists of 730 sequences with 267 unique symbols, and this dataset also contained some of the longest sequences found among all datasets we explored. The dataset was created by the National Center for Sign Language and Gesture Resources at Boston University. Each sequences represents an “utterance” consisting of ASL gestures and facial expressions [21]. Due to the length of the sequences and relatively few symbols, the SIGN dataset required the largest amount of time to extract prefix-closed patterns. The result seen in Fig. 6 shows the time required to mine the complete set of prefix-closed patterns when every possible single item suffix is used. Despite an identical search space explored by both algorithms, Algorithm 4 still outperforms Algorithm 3. This is most likely because the subsumption checking used in Algorithm 3 is more computationally expensive than projected database creation used in Algorithm 4.



All Possible Single Item Suffixes

Fig. 6. SIGN dataset

6 Conclusions and Future Work

In this paper, we introduced the problems of frequent prefix mining, prefix-closed mining, and prefix-maximal mining along with their suffix variants. We have shown the usefulness of mining projected databases for obtaining prefix/suffix patterns and have proven that these approaches produce the complete set of frequent prefix/suffix patterns. Theoretical analysis shows that it is better to create multiple projected databases when faced with multiple prefixes/suffixes of interest (as apposed to mining the original database a single time), and empirical analysis supports this conclusion. Empirical analysis also shows that our proposed algorithms, while not more efficient in the sense of being in a better complexity class, tend to be an order of magnitude faster in practice.

In the future, we want to explore the use of prefix/suffix sequential pattern mining for predicting health trajectories of cancer treatments. By mining suffix patterns related to cancer treatments, we hope to develop probabilistic rules that link particular cancer treatments to subsequent health trajectories. In addition, we want to leverage prefix pattern mining in the hopes that prior medical history will allow us to better predict health trajectories after a particular treatment.

References

1. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the Eleventh International Conference on Data Engineering (1995)
2. Huang, C.L., Huang, W.L.: Handling sequential pattern decay: developing a two-stage collaborative recommender system. *Electr. Commer. Res. Appl.* **8**, 117–129 (2009)
3. Yap, G.-E., Li, X.-L., Yu, P.S.: Effective next-items recommendation via personalized sequential pattern mining. In: Lee, S., et al. (eds.) DASFAA 2012. LNCS, vol. 7239, pp. 48–64. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29035-0_4
4. Baralis, E., et al.: Analysis of medical pathways by means of frequent closed sequences. In: Setchi, R., Jordanov, I., Howlett, R.J., Jain, L.C. (eds.) KES 2010. LNCS (LNAI), vol. 6278, pp. 418–425. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15393-8_47
5. Uragaki, K., Hosaka, T., Arahori, Y., Kushima, M., Yamazaki, T., Araki, K., Yokota, H.: Sequential pattern mining on electronic medical records with handling time intervals and the efficacy of medicines. In: 2016 IEEE Symposium on Computers and Communication (ISCC) (2016)
6. Aloysius, G., Binu, D.: An approach to products placement in supermarkets using PrefixSpan algorithm. *J. King Saud Univ.-Comput. Inf. Sci.* **25**, 77–87 (2013)
7. Shim, B., Choi, K., Suh, Y.: Crm strategies for a small-sized online shopping mall based on association rules and sequential patterns. *Expert Syst. Appl.* **39**, 7736–7742 (2012)
8. Chen, Y.L., Hu, Y.H.: Constraint-based sequential pattern mining: the consideration of recency and compactness. *Decis. Support Syst.* **42**, 1203–1215 (2006)
9. Antunes, C., Oliveira, A.L.: Generalization of pattern-growth methods for sequential pattern mining with gap constraints. In: Perner, P., Rosenfeld, A. (eds.) MLDM 2003. LNCS, vol. 2734, pp. 239–251. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45065-3_21
10. Li, C., Wang, J.: Efficiently mining closed subsequences with gap constraints. In: Proceedings of the 2008 SIAM International Conference on Data Mining (2008)
11. Srikant, R., Agrawal, R.: Mining sequential patterns: generalizations and performance improvements. In: Apers, P., Bouzeghoub, M., Gardarin, G. (eds.) EDBT 1996. LNCS, vol. 1057, pp. 1–17. Springer, Heidelberg (1996). <https://doi.org/10.1007/BFb0014140>
12. Kaytoue, M., Pitarch, Y., Plantevit, M., Robardet, C.: What effects topological changes in dynamic graphs? Elucidating relationships between vertex attributes and the graph structure. *Soc. Netw. Anal. Min.* **5**, 55 (2015)
13. Zaki, M.J.: SPADE: an efficient algorithm for mining frequent sequences. *Mach. Learn.* **42**, 31–60 (2001)
14. Ayres, J., Flannick, J., Gehrke, J., Yiu, T.: Sequential pattern mining using a bitmap representation. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2002)
15. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.C.: PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth. In: Proceedings of the 17th International Conference on Data Engineering (2001)
16. Wang, J., Han, J.: BIDE: efficient mining of frequent closed sequences. In: Proceedings of the 20th International Conference on Data Engineering (2004)

17. Yan, X., Han, J., Afshar, R.: CloSpan: mining closed sequential patterns in large datasets. In: Proceedings of the 2003 SIAM International Conference on Data Mining (2003)
18. 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**, 54–77 (2017)
19. 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
20. 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. Customer Strategy Manag.* **19**, 197–208 (2012)
21. Neidle, C.: SignStreamTM: a database tool for research on visual-gestural language. *Sign Lang. Linguist.* **4**, 203–214 (2001)

Author Index

- Ahmed, Chowdhury Farhan 29, 44, 59, 199, 215
Ahmed, Hosneara 44
Alam, Tahira 59
Alaminos, David 296
Aman, Rutba 199
Ananthakumar, Usha 1
Arefin, Mohammad Fahim 29
- Batko, Michal 183
- Costa, Hugo 75
- Dai, Bi-Ru 240
Du, Xiao-Lin 119
- Eberle, William 309
Efremova, Julia 288
Endres, Ian 288
Enomoto, Shota 135
- Fernández, Manuel A. 296
Fernández, Sergio M. 296
Ferreira, Eija 17
- García, Francisca 296
Gelbard, R. 173
Graves, Jeffrey A. 309
- He, Junyi 104
Helaakoski, Heli 17
- Ishita, Sabrina Zaman 215
Islam, Maliha Tashfia 29, 59
Isobe, Takashi 266
- Jokisaari, Juha 17
Jones, Paul 104
- Khalemsky, A. 173
Khan, Muhammad Asif Hossain 59
Kohara, Kazuhiro 135
Kyllönen, Vesa 17
- Le, Bac 272
- Melnik, Ofer 288
- Nalepa, Filip 183
Ni, Haoqi 104
Noor, Faria 215
- Okada, Yoshihiro 266
- Paja, Wiesław 230
Phan, Huan 272
Puukko, Esa 17
- Qiu, Xiaofeng 254
- Ritter, Marc 148
Rocha, Miguel 75
Rodrigues, Ruben 75
- Samatova, Nagiza 104
Shahee, Shaukat Ali 1
Singh, Rina 309
Song, Sherraina 88
Sultana, Abeda 44
- Talbert, Douglas A. 309
Tamminen, Satu 17
Tang, Yi 240
Tiensuu, Henna 17

Vidas, Isaac 288
Vodel, Matthias 148

Wang, Dan 119
Wang, Meng 119

Xie, Ying 162
Xu, Mingyang 104
Xuan, De 162

Yang, Ping 119
Yang, Ruixin 104
Yi, Xiu 162

Zahin, Sabit Anwar 59
Zezula, Pavel 183
Zhao, Tianyue 254
Zhong, Junmei 162