Proceedings of the Workshops of the EDBT/ICDT 2018 Joint Conference (EDBT/ICDT 2018)

Nikolaus Augsten
University of Salzburg
Austria
Vienna, Austria, March 26, 2018
1 Message from the Chairs

2 International Workshop on Data Analytics Solutions for Real-Life Applications

DARLI-AP
2.1 Community Detection and Correlated Attribute Cluster Analysis on Multi-Attributed Graphs ................................................. 2
2.2 Classifying Land Cover from Satellite Images Using Time Series Analytics ................................................................. 10
2.3 Data Analytics to Improve Co-Operative Education ................................................................. 16
2.4 Recommendation of Job Offers Using Random Forests and Support Vector Machines ................................................................. 22
2.5 Sensitive Attribute Prediction for Social Networks Users ................................................................. 28
2.6 MLNET - Machine Learning Models for Network Analytics ................................................................. 36
2.7 Unsupervised Video Semantic Partitioning Using IBM Watson And Topic Modelling ................................................................. 44
2.8 Study of the Applicability of an Itemset-Based Portfolio Planner in a Multi-Market Context ................................................................. 50
2.9 Mining Implicit Data Association from TripAdvisor Hotel Reviews ................................................................. 56
2.10 Using a NoSQL Graph Oriented Database to Store Accessible Transport Routes ................................................................. 62
2.11 Guided Query Composition with Semantic OLAP Patterns ................................................................. 67

3 International Workshop on Big Data Visual Exploration and Analytics

BigVis
3.1 Enabling Hierarchical Exploration for Large-Scale Multidimensional Data with Abstract Parallel Coordinates ................................................................. 76
3.2 Finding the Needle in a Haystack: Entropy Guided Exploration of Very Large Graph Cubes ................................................................. 84
3.3 Map-Based Visual Exploration of Geolocated Time Series ................................................................. 92
3.4 An Interactive 3D Visualization for the LOD Cloud ................................................................. 100
3.5 Topic-aware Network Visualisation to Explore Large Email Corpora ................................................................. 104

4 Big Mobility Data Analytics

BMDA
4.1 A Distributed Online Learning Approach for Pattern Prediction over Movement Event Streams with Apache Flink ................................................................. 109
4.2 Mining Vessel Trajectory Data for Patterns of Search and Rescue ................................................................. 117
4.3 DiStRDF: Distributed Spatio-temporal RDF Queries on Spark ................................................................. 125
4.4 Short-Term Traffic Forecasting: A Dynamic ST-KNN Model Considering Spatial Heterogeneity and Temporal Non-Stationarity ................................................................. 133
Message from the Chairs

On behalf of the entire conference organizing committee and the workshop organizers we are delighted to present the proceedings of the Workshops of the EDBT/ICDT 2018 Joint Conference, held on March 26, 2018, in Vienna, Austria.

The International Conference on Extending Database Technology (EDBT) and the International Conference on Database Theory (ICDT) are two prestigious forums for the exchange of the latest research results in data management and the theoretical foundations of database systems. While having the same overarching goal of presenting cutting-edge results, ideas, techniques, and theoretical advances in databases, the workshops of the EDBT/ICDT joint conference are separately tasked by focusing on emerging topics, complementing the areas covered by the main technical program.

This volume covers the International Workshop on Data Analytics Solutions for Real-Life Applications (DARLI-AP), the International Workshop on Big Data Visual Exploration and Analytics (BigVis), and the workshop Big Mobility Data Analytics (BMDA). The proceedings of the workshop Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP), which was co-located with EDBT/ICDT 2018 as well, are published in a separate volume [1].

Data Analytics Solutions for Real-Life Applications (DARLI-AP) DARLI-AP aims at promoting and sharing research and innovation on data analytics solutions/strategies for real-life and cutting-edge applications. The use of Information and Communication Technologies has made a huge amount of heterogeneous data available in various real application domains (e.g., smart cities, health care systems, financial applications, banking and insurance, Industry 4.0). A data scientist is required to tackle the no-trivial task of selecting the best techniques to effectively and efficiently deal with issues related to storage, search, sharing, modeling, analysis, and visualization of data, information, and knowledge. The complexity of the task increases with variable data distributions, data heterogeneity, and data volume. Furthermore, a rich spectrum of knowledge can be extracted from the data to characterize user behaviors, improve the quality of provided services, or even devise new ones, thus increasing the benefits of real-life applications.

DARLI-AP allows academics and practitioners from various research areas to share their experiences on designing cutting-edge analytics solutions for real-life applications. Researchers are encouraged to present their work-in-progress research activity describing innovative methodologies, algorithms, or platforms addressing all facets of the data analytics process. Also industrial implementations of data analytics applications as well as design and deployment experience reports are welcome.

Big Data Visual Exploration and Analytics (BigVis) One of the major challenges of the Big Data era is the availability of a great amount and variety of massive datasets to be analyzed by non-corporate data analysts such as research scientists, data journalists, policy makers, SMEs, and individuals. A major characteristic of these datasets is that they are: accessible in a raw format that is not being loaded or indexed in a database (e.g., plain text files, json, rlf), dynamic, dirty, and heterogeneous in nature. Data-curious users who would like to access and analyze these datasets face great challenges that are even more burdensome for the increasing number of non-expert users. The purpose of visual data exploration is to facilitate information perception and manipulation, knowledge extraction, and inference by non-expert users. In the Big Data era, several
challenges arise in the field of data visualization and analytics. First, modern exploration and visualization systems should offer scalable data management techniques in order to efficiently handle billion objects datasets, limiting the system response to a few milliseconds. Besides, systems must address the challenge of on-the-fly scalable visualizations over large and dynamic sets of volatile raw data, offering efficient interactive exploration techniques, as well as mechanisms for information abstraction, sampling, and summarization for addressing problems related to information over-plotting. Further, they must encourage user comprehension offering customization capabilities to different user-defined exploration scenarios and preferences according to the analysis needs. Overall, the challenge is to enable users to gain value and insights out of the data as rapidly as possible, minimizing the role of IT-experts in the loop.

The BigVis workshop aims at addressing the above challenges and issues by providing a forum for researchers and practitioners to discuss, exchange, and disseminate their work. BigVis addresses the research areas of Data Management and Mining, Information Visualization, and Human-Computer Interaction, and encourages novel works that establish ties between these communities.

Big Mobility Data Analytics (BMDA) Nowadays, we have the means to collect, store, and process mobility data of an unprecedented quantity, quality, and timeliness. This is mainly due to the wide spread of GPS-equipped devices, including new generation smartphones. As ubiquitous computing pervades our society, mobility represents a very useful source of information. Movement traces, especially when combined with societal data, can aid transportation engineers, urban planners, and eco-scientists towards decision making in a wide spectrum of applications, such as traffic engineering and risk management.

The objective of BMDA is to bring together researchers and practitioners interested in scalable data-intensive applications that manage and analyze big mobility data. The workshop fosters the exchange of new ideas on multidisciplinary real-world problems, the discussion on proposals about innovative solutions, and the identification of emerging research opportunities in the area of big mobility data analytics. Thereby, all layers of the Big Data Value Analytics reference model are of interest, namely data management, data processing, data analytics, data visualization, and user interaction. BMDA intends to bridge the gap between researchers and big data stakeholders, including experts from critical domains, such as urban / maritime / aviation transportation or human complex networks. Most importantly it aims at unveiling real-world problems and depicting novel solutions in such domains that require innovative data analytics solutions.

We would like to acknowledge those who have contributed to the success of the workshop program. We thank the workshop chairs for their efforts in organizing the workshops and for putting together an exciting program, and the PC members and external reviewers for their invaluable contribution. We also thank the invited speakers for enriching the workshop programs, the authors for continuing to submit their high-quality work to the EDBT/ICDT workshops, and the conference organizers and volunteers for the realization of this event. Finally, we would like to acknowledge the technical support of Manuel Widmoser with the proceedings.

Nikolaus Augsten, Workshop Chair
Michael Böhlen, EDBT Program Chair
Benny Kimelfeld, ICDT Program Chair
Reinhard Pichler, General Chair
References

International Workshop on Data Analytics Solutions for Real-Life Applications

DARLI-AP

Tania Cerquitelli, Politecnico di Torino, Italy
Silvia Chiusano, Politecnico di Torino, Italy
Genoveva Vargas-Solar, CNRS, LIG-LAFMIA, France
Community Detection and Correlated Attribute Cluster Analysis on Multi-Attributed Graphs

Hiroyoshi Ito†, Takahiro Komamizu‡, Toshiyuki Amagasa‡, and Hiroyuki Kitagawa‡

†Graduate School of Systems and Information Engineering, University of Tsukuba
‡Center for Computational Sciences, University of Tsukuba
hiro.3188@kde.cs.tsukuba.ac.jp, taka-coma@acm.org, {amagasa,kitagawa}@cs.tsukuba.ac.jp

ABSTRACT
Multi-attributed graphs, in which each node is characterized by multiple types of attributes, are ubiquitous in the real world. Detection and characterization of communities of nodes could have a significant impact on various applications. Although previous studies have attempted to tackle this task, it is still challenging due to difficulties in the integration of graph structures with multiple attributes and the presence of noises in the graphs. Therefore, in this study, we have focused on clusters of attribute values and strong correlations between communities and attribute-value clusters. The graph clustering methodology adopted in the proposed study involves Community detection, Attribute-value clustering, and deriving Relationships between communities and attribute-value clusters (CAR for short). Based on these concepts, the proposed multi-attributed graph clustering is modeled as CAR-clustering. To achieve CAR-clustering, a novel algorithm named CARNMF is developed based on non-negative matrix factorization (NMF) that can detect CAR in a cooperative manner. Results obtained from experiments using real-world datasets show that the CARNMF can detect communities and attribute-value clusters more accurately than existing comparable methods. Furthermore, clustering results obtained using the CARNMF indicate that CARNMF can successfully detect informative communities with meaningful semantic descriptions through correlations between communities and attribute-value clusters.

1 INTRODUCTION
Community detection is a task to detect densely connected subgraphs as communities. Nodes in a community tend to share same or similar properties, such phenomenon is called homophily effect [11, 17], meaning that nodes having similar properties tend to link together. Because diverse applications are derived from the nature of real communities, community detection is important in graph/network analyses. Examples include node property estimations [7, 9, 24], community-wise information recommendations [10], and semantic reasoning for nodes/edges [1].

Moreover, using the attributes in a graph is advantageous to realize high-quality community detection as well as to understand the characteristics of communities. Multi-attributed graphs are reasonable models of real-world networks such as social networks, co-author networks, protein-protein interaction networks, etc. In fact, several works have proposed algorithms that employ attribute information (i.e., shared interests or functional behaviors of each community) to detect not only communities but also their semantic meanings [19, 23, 25, 26]. However, community detection and extraction of semantics in multi-attributed graphs remain challenging due to difficulties in integrating graph structures and multiple attributes of different types. Community detection and extraction of semantics involves multiple steps. First, useful information from each attribute must be extracted because certain node attributes describe different aspects. Second, all extracted information must be exploited to enhance community detection by effectively integrating heterogeneous information. Notice that the previous works [19, 23, 25, 26] do not differentiate multiple attributes, that is, they consider multiple attributes equally. Moreover, real-world graphs are often incomplete and noisy. That is, some edges or nodes may be missing or attribute values may contain incorrect values, leading to inappropriate results.

To overcome these difficulties, we propose a novel clustering scheme based on the following two assumptions:
(1) Relevant attribute values form clusters by attribute type. This is based on the observation that an attribute reflects a node’s interests in a network. Hence, an attribute tends to be associated to a specific group of values related to an interest. For example, in a co-author network where the nodes correspond to the authors (researchers), each author typically has specific research interests (e.g., AI, data mining, and database). Thus, attributes (e.g., paper title and conference) present biased values according to interests. Consequently, it is possible to identify clusters of attributes values (attribute-value clusters) reflecting a node’s interests.
(2) Communities are strongly correlated with attribute-value clusters. This is related to the previous assumption. Consider the example above. The nodes in a community share similar interests (e.g., research interests) and consequently, similar attribute-value clusters (e.g., research topics, and conferences). Conversely, if some nodes (researchers) have similar attribute values, they should share similar interests and can be grouped in the same community.

Exploiting the correlation between communities and multiple attributes should improve the quality of community detection as well as attribute-value clustering. Using the information from different sources (attributes) to alleviate the effect of noise (e.g., missing values and errors), we simultaneously implement community detection and attribute-value clustering.

Based on the aforementioned ideas, we study a novel clustering scheme for multi-attributed graphs, called CAR-clustering. CAR includes Community detection, Attribute-value clustering, and deriving Relationships between communities and attribute-value clusters for multi-attributed graphs. Additionally, we develop a novel clustering algorithm called CARNMF, which employs a non-negative matrix factorization (NMF).

The contributions of this paper are summarized as follows:
• We propose a novel clustering scheme CAR-clustering to address two technical questions. (i) Given a multi-attributed
graph, how can community detection and attribute-value clustering be performed for different types of attributes in a cooperative manner? (ii) How should reasonable relationships be determined between communities and attribute-value clusters for each type of attribute?

- We develop a novel algorithm CARNMF, which achieves CAR-clustering. Specifically, a dedicated loss function is designed to perform multiple NMFs simultaneously.
- We conduct experiments using real-world datasets (DBLP computer science bibliography and arXiv physics bibliography). The accuracy of CARNMF with respect to community detection and attribute-value clustering and a comparison to other methods are examined. Relative to comparative methods, CARNMF achieves a better accuracy of up to 11% for community detection and up to 22% for attribute-value clustering. Furthermore, CARNMF detects informative communities and their rich semantic descriptions by correlating multiple types of attribute-value clusters.

2 RELATED WORK

Community detection in graphs is a current topic of interest in graph analysis and AI research. Existing works for non-attributed graphs can be categorized according to the techniques used: graph separation [9, 20], probabilistic generative model [27], and matrix factorization [12, 18, 24]. [9] defined modularity, which indicates how separated a community is from other nodes. More comprehensive surveys can be found in [6, 22].

Recently, several works have addressed the problem of detecting communities and their semantic descriptions on node-attributed graphs. [25] proposed CESNA, where communities and their attributes are simultaneously detected in an efficient manner. [23] proposed SCI to detect communities and their semantics using NMF. [19] proposed a probabilistic generative model called the author-topic model to model communities and related topics. [2] proposed COMODO to detect communities with shared properties using subgroup discovery techniques. Likewise, [26] proposed LCTA, where communities and their topics are modeled separately, and then their relationships are modeled using a probabilistic generative model. A comprehensive survey over these works can be found in [5].

The aforementioned works only consider single textual attributes or uniformly handle multiple attributes without any distinction. In reality, each attribute represents different aspects of the nodes. In our research, we deal with heterogeneous attributes individually. In addition to community detection, we perform clustering over attribute values for each attribute, which, in turn, can be used to improve the quality of communities detected.

Some works have investigated clustering over networks containing different types of nodes and/or edges. [3] studied community detection with characterization from multidimensional networks, which is defined as a graph consisting of a set of nodes and multiple types of edges. [4] studied subgraph detection from multi-layer graphs with edge labels. In contrast, we assume a different model where each node is characterized by multiple attributes. [21] proposed a scheme of ranking-based clustering for multi-typed heterogeneous networks, where two or more types of nodes are included. Similarly, [16] proposed an NMF-based method for such networks. These methods differ from ours in that they define a cluster consisting of all types of nodes. In other words, these methods cannot handle each attribute in a unique way. In contrast, our work deals with different attributes individually, but solves community detection and attribute-value clustering in a unified manner.

3 PROBLEM STATEMENT

In this work, we deal with multi-attributed graphs, where each node is characterized by two or more attributes. Given such a graph, CAR-clustering is used to solve the following three sub-problems: community detection, attribute-value clustering, and derivation of relationships between communities and attribute-value clusters, which have been independently studied. Below, we provide the formal definitions which are necessary to define the clustering scheme.

3.1 Multi-Attributed Graph

Multi-attributed graph \( G \) is defined by extending weighted graph \( G' \) with several attributed graphs \( G_t \) for attribute \( t \in T \). The following are formal definitions.

**Definition 1 (Weighted graph).** Weighted graph \( G' \) is defined by a triplet \((\mathcal{V}, \mathcal{E}, \mathcal{W})\), where \( \mathcal{V} \) is a set of nodes, \( \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \) is a set of edges, and \( \mathcal{W} : \mathcal{E} \rightarrow \mathbb{R}^+ \) is a map of edge weights.

**Definition 2 (Attributed graph).** Attributed graph \( G_t = (\mathcal{V} \cup \mathcal{X}_t, \mathcal{E}_t, \mathcal{W}_t) \) of attribute \( t \in T \) is a bipartite graph consisting of set \( \mathcal{V} \) of nodes, set \( \mathcal{X}_t \) of attribute-values, a set of edges \( \mathcal{E}_t \subseteq \mathcal{V} \times \mathcal{X}_t \), and \( \mathcal{W}_t : \mathcal{E}_t \rightarrow \mathbb{R}^+ \) is a map of edge weights.

**Definition 3 (Multi-attributed graph).** Given weighted graph \( G' = (\mathcal{V}, \mathcal{E}, \mathcal{W}) \) and a set of attributed graphs \( G_t \) where \( G_t = (\mathcal{V} \cup \mathcal{X}_t, \mathcal{E}_t, \mathcal{W}_t) \), multi-attributed graph \( G = (G', (G_t)_{t \in T}) \) is a union of these graphs.

3.2 CAR-clustering

Given a multi-attributed graph, information can be extracted from different perspectives. In this work, we extract communities, attribute-value clusters, and the relationship between them.

**Community.** For a multi-attributed graph, a set of nodes with the following properties is regarded as a community. (1) Nodes in a community are densely connected with each other and sparsely connected with other nodes. (2) Nodes in a community tend to share common values in distinct attributes. This study assumes that communities can overlap. That is, each node belongs to more than one community. This assumption is reasonable for real applications. Formally, given the number of communities \( \ell \), node \( n \in \mathcal{V} \) belonging to community \( c \in \mathcal{C} \) is described by probability distribution \( p(n | c) \), where \( |\mathcal{C}| = \ell \).

**Attribute-value cluster.** For attribute \( t \in T \) in a multi-attributed graph, similar or highly correlated attribute values can be grouped into attribute-value clusters. Herein, we assume overlapping clusters. That is, each attribute-value belongs to more than one cluster. Formally, given the number of clusters \( k_t \) of attribute \( t \in T \), cluster member \( x \in \mathcal{X}_t \) for attribute-value cluster \( s_t \in \mathcal{S}_t \) is described by probability distribution \( p(x | s_t) \), where \( |\mathcal{S}_t| = k_t \).

**Relationship between a community and an attribute-value cluster.** Nodes in a community often share common attribute-value clusters. Detecting such relationship is useful in many applications. Given community \( c \in \mathcal{C} \) and attribute-value cluster \( s_t \in \mathcal{S}_t \) of attribute \( t \in T \), the probability that \( c \) is related to \( s_t \) is described as the relationship between \( c \) and \( s_t \). In this work, a community may be related to more than one attribute-value cluster. Formally, this is described by probability distribution \( p(s_t | c) \).
CAR-clustering. CAR-clustering is formally defined by Definition 4.

**Definition 4 (CAR-clustering).** Given a multi-attributed graph \( G \), CAR-clustering is to perform community detection, attribute-value clustering, and detection of the relationship between the communities and the attribute-value clusters simultaneously.

Solving these sub-problems simultaneously is more beneficial than evaluating each one independently because, in many cases, communities and attribute-value clusters are mutually correlated. Solving the problems simultaneously exploits this correlation, leading to improved results.

4 CAR-NMF – ALGORITHM FOR CAR-CLUSTERING

In this section, we propose an NMF (non-negative matrix factorization)-based algorithm, called CAR-NMF, for CAR-clustering. CAR-NMF models communities and attribute-value clusters. Additionally, we introduce an auxiliary matrix to maintain the relationship between the communities and the attribute-value clusters. A unified loss function is used to solve the different NMFs in a unified manner. It is assumed that the user given the number \( \ell \) of communities and the number \( k_t \) of clusters for each attribute \( t \in \mathbb{T} \).

4.1 Matrix representation

We represent a multi-attributed graph by two sorts of matrices: an adjacency matrix \( A \in \mathbb{R}^{[V] \times |X_T|} \) and attribute matrices \( X_t \in \mathbb{R}^{[V] \times |X_T|} \) for \( t \in \mathbb{T} \). An element \( A_{u,v} \) of \( A \) corresponds to an edge \( e_{u,v} = (u,v) \in \mathbb{E} \). An element \( X_t^{(e_{u,v})} \) of \( X_t \) corresponds to an edge \( e_{u,v} \in \mathbb{E}_t \). The cell contains the joint probability of the presence of edge \( e_{u,v} \).

4.2 Loss Function

We achieve CAR-clustering in terms of several NMFs, which correspond to the aforementioned sub-problems. To achieve CAR-clustering, we introduce loss functions for the sub-problems followed by a unified loss function.

**Loss function for community detection.** In CAR-NMF, communities \( C \) are denoted by a matrix \( U^* \in \mathbb{R}^{[V] \times |C|} \), where each row and column correspond to a node \( u \in V \) and a community \( c \in C \), respectively. A cell \( U^*_{u,c} \) represents probability \( p(u \mid c) \). In addition, joint probability \( p(u,v \mid c) \), \( u \) and \( v \) are connected through community \( c \), and is represented by \( U^*_{u,v} \). Moreover, joint probability \( p(u,v) \), or the existence of edge \( e_{u,v} \in \mathbb{E} \), is expressed as \( \sum_{c \in C} U^*_{u,v} U^*_{v,c} \). Therefore, when \( U^* \) minimizes the following loss function, \( U^* \) is the best approximation of the edges in the graph.

\[
\arg \min_{U^* \geq 0} \left\| A - U^*(U^*)^T \right\|_F^2 \\
\text{s.t.} \ 1 \leq c \leq \ell, \ \left\| U^*_{:,c} \right\|_1 = 1
\]

where \( \| \cdot \|_F \) and \( \| \cdot \|_1 \) represents the Frobenius norm and the \( \ell_1 \) norm, respectively.

**Loss function for attribute-value clustering.** In CAR-NMF, attribute-value clusters \( S_t \) of attribute \( t \in \mathbb{T} \) are represented as a matrix \( V^{(t)} \in \mathbb{R}^{[X_T] \times k_t} \), where each row and column correspond to an attribute \( x \in X_T \) and an attribute cluster \( s_r \in S_t \), respectively. A cell \( V^{(t)}_{x,s_r} \) represents probability \( p(x \mid s_r) \).

To derive \( V^{(t)} \) from \( X_t^{(e_{u,v})} \), we introduce a matrix \( U_t \in \mathbb{R}^{[V] \times k_t} \), which denotes the relationships between the nodes and attribute-value clusters with probability \( p(u \mid s_r) \). Using both matrices \( U_t \) and \( V^{(t)} \), probability \( p(u,x \mid s_r) \), which is the existence of edge \( e_{u,x} \in \mathbb{E}_t \) in terms of attribute-value cluster \( s_r \), is calculated as \( U_t^{(e_{u,x})} V^{(t)}_{s_r,x} \). Moreover, probability \( p(u,x) \) is derived as \( \sum_{s_r \in S_t} U_t^{(e_{u,x})} V^{(t)}_{s_r,x} \). Therefore, when \( U_t \) and \( V^{(t)} \) minimize loss function, \( f_t(U_t, V^{(t)}) \) represent the best approximation of the edges in the graph.

\[
\arg \min_{U_t, V^{(t)} \geq 0} \left\| X_t^{(e_{u,v})} - (U_t^{(e_{u,v})} V^{(t)})^T \right\|_F^2 \\
\text{s.t.} \ 1 \leq t \leq \ell, \ \left\| V^{(t)} \right\|_F = 1
\]

**Loss function for relationship detection.** In CAR-NMF, the relationships between communities and attribute-value clusters of attribute \( t \in \mathbb{T} \) are represented as a matrix \( G^{(t)} \in \mathbb{R}^{[C] \times k_t} \), where each row and column corresponds to a community \( c \in C \) and an attribute-value cluster \( s_r \in S_t \), respectively. The cell contains the power \( p(s_r \mid c) \). We assume \( R^{(t)} \) is a linear transformation that maps \( U^* \) into \( U_t \), where \( U^* \) and \( U_t^{(e_{u,v})} \) derived by Equation 1 and Equation 2, respectively. Therefore, when \( R^{(t)} \) minimizes the loss function, \( R^{(t)} \) represents the relationships between the communities and the attribute-value clusters.

\[
\arg \min_{U_t, V^{(t)}, R^{(t)} \geq 0} \left\| U_t^{(e_{u,v})} - U^* R^{(t)} \right\|_F^2 \\
\text{s.t.} \ 1 \leq t \leq \ell, \ V_{:,p}^T R^T_{:,p} = 1, \ \left\| R^{(t)} \right\|_1 = 1
\]

Equation 3 can be regarded as an NMF that decomposes the matrix of the node-by-attribute value cluster into node-by-community and community-by-attribute value cluster matrices. In other words, Equation 3 indicates the effect of the relationship between nodes and attribute-value clusters against communities.

**Unified loss function.** To achieve CAR-clustering, the aforementioned three sub-problems must be solved. In this work, we attempt to solve them simultaneously by introducing a unified loss function, which is expressed as

\[
L = \arg \min_{U_t, \{U^{(t)}, V^{(t)}, R^{(t)}\}_{t \in \mathbb{T}}} \left\| A - U^*(U^*)^T \right\|_F^2 \\
+ \sum_{t \in \mathbb{T}} \left( \left\| X_t^{(e_{u,v})} - (U_t^{(e_{u,v})} V^{(t)})^T \right\|_F^2 + \lambda_t \left\| U_t^{(e_{u,v})} - U^* R^{(t)} \right\|_F^2 \right) \\
\text{s.t.} \ 1 \leq t \leq \ell, \ V_{:,p}^T R^T_{:,p} = 1, \ V_{:,p}^T R^T_{:,p} = 1, \ \left\| R^{(t)} \right\|_1 = 1 \]

where \( \lambda_t \) for attribute \( t \in \mathbb{T} \) is a user-defined parameter to control the effect of attribute-value clusters for community detection. Higher \( \lambda_t \) yields a stronger effect of the attribute-value clusters in community detection.

4.3 Optimization

Similar to the ordinary NMF, the loss function in Equation 4 is not simultaneously convex for all variables. Hence, we consider the NMF to be a Frobenius norm optimization, where update equations are derived based on [14].
Considering the Karush-Kuhn-Tucker (KKT) first-order conditions applied to our problem, we derive:

\[ U^* \geq 0, \quad U^{(t)} \geq 0, \quad V^{(t)} \geq 0, \quad R^{(t)} \geq 0 \]  
\[ \nabla U^* \leq 0, \quad \nabla U^{(t)} \leq 0, \quad \nabla V^{(t)} \leq 0, \quad \nabla R^{(t)} \leq 0 \]  
\[ U^* \odot \nabla U^* = 0, \quad U^{(t)} \odot \nabla U^{(t)} = 0, \quad V^{(t)} \odot \nabla V^{(t)} = 0, \quad R^{(t)} \odot \nabla R^{(t)} = 0 \]

where \( \odot \) is the element-wise product. From the Karush-Kuhn-Tucker (KKT) conditions, we derive derivatives corresponding to the variables:

\[ \nabla U^* = -2A^T R U^* + 2U^* (U^*)^T U^* + \sum_{t \in T} \lambda_t (U^{(t)} (R^{(t)})^T + U^* (R^{(t)})^T) \]  
\[ \nabla U^{(t)} = -X^{(t)} V^{(t)} + U^{(t)} (V^{(t)})^T V^{(t)} + \lambda_t (U^{(t)} - U^{(t)}) \]  
\[ \nabla V^{(t)} = -(X^{(t)})^T U^{(t)} + (V^{(t)})^T U^{(t)} + (V^{(t)})^T U^{(t)} \]  
\[ \nabla R_t = -U^{(t)} (U^{(t)})^T + (U^*)^T U^{(t)} \]

By substituting the corresponding gradients in Equation 4, we derive the following update rules:

\[ U^* \leftarrow U^* - 2A^T R U^* + 2U^* (U^*)^T U^* + \sum_{t \in T} \lambda_t (U^{(t)} (R^{(t)})^T + U^* (R^{(t)})^T) \]  
\[ U^{(t)} \leftarrow U^{(t)} + \frac{X^{(t)} V^{(t)} + U^{(t)} (V^{(t)})^T V^{(t)} + \lambda_t U^{(t)} (R^{(t)})^T}{U^{(t)} (V^{(t)})^T V^{(t)} + \lambda_t U^{(t)}} \]  
\[ V^{(t)} \leftarrow V^{(t)} - \frac{(X^{(t)})^T U^{(t)} + (V^{(t)})^T U^{(t)} + (V^{(t)})^T U^{(t)}}{(X^{(t)})^T U^{(t)}} \]  
\[ R^{(t)} \leftarrow R^{(t)} - \frac{U^{(t)} U^{(t)} + (U^*)^T U^{(t)}}{(U^*)^T U^{(t)}} \]

The aforementioned update rules monotonically decrease Equation 4. However, these variables may violate the probability definition (i.e., their sum does not equal one). To satisfy constraints, \( \| U^* \|_1 = 1, \| V^{(t)} \|_1 = 1 \) and \( \| R^{(t)} \|_1 = 1 \), the variables are normalized immediately after updating. The normalized as follows:

\[ U^* \leftarrow Q^{(t)} \]  
\[ U^{(t)} \leftarrow U^{(t)} Q^{(t)} \]  
\[ V^{(t)} \leftarrow V^{(t)} Q^{(t)} \]  
\[ R^{(t)} \leftarrow R^{(t)} Q^{(t)} \]

where \( Q^{(t)} = \text{Diagonalize}(U^*), Q^{(t)} = \text{Diagonalize}(V^{(t)}), \) and \( Q^{(t)} = \text{Diagonalize}(R^{(t)}) \).

\[ \text{Diagonalize}(Z \in \mathbb{R}^{a \times b}) = \text{Diag} \left( \sum_{i=1}^{a} Z_{i,1} \cdots \sum_{i=1}^{a} Z_{i,b} \right) \]

Algorithm 1 Optimization Algorithm

Input: \( A, \{X^{(t)}\}_{t \in T}, \{\lambda_t\}_{t \in T}, \delta \)
Output: \( U^*, \{U^{(t)}, V^{(t)}, R^{(t)}\}_{t \in T} \)
1. \( U^*, \{U^{(t)}, V^{(t)}, R^{(t)}\}_{t \in T} \leftarrow \text{random non-negative init} \)
2. \( \epsilon' \leftarrow \text{maxFloat}, \epsilon \leftarrow \frac{\delta}{2} \)
3. while \( \text{abs}(\epsilon' - \epsilon) \geq \delta \) do
4. \( U^* \leftarrow U^* - \frac{A^T U^* + \sum_{t \in T} \lambda_t U^{(t)} (R^{(t)})^T}{\|U^*\|_F + \sum_{t \in T} \lambda_t U^{(t)} (R^{(t)})^T} \)
5. \( U^{(t)} \leftarrow U^{(t)} - \frac{X^{(t)} V^{(t)} + \lambda_t U^{(t)} (R^{(t)})^T}{\|U^{(t)}\|_F + \lambda_t U^{(t)}} \)
6. for \( t \in T \) do
7. \( V^{(t)} \leftarrow V^{(t)} - \frac{(X^{(t)})^T U^{(t)} + \lambda_t U^{(t)} (R^{(t)})^T}{\|X^{(t)}\|_F} \)
8. \( R^{(t)} \leftarrow R^{(t)} - \frac{U^{(t)} U^{(t)} + \lambda U^{(t)}}{(U^{(t)})^T U^{(t)}} \)
9. \( \epsilon' \leftarrow \epsilon \)
10. \( \epsilon \leftarrow \frac{L}{2} \left( U^*, \{U^{(t)}, V^{(t)}, R^{(t)}\}_{t \in T} \right) \)
end while

4.4 Complexity Analysis

Here, we analyze the computational complexity of the proposed algorithm. The equations in our algorithm have the following complexities:

- Updating \( U^* \) (Eqs. 12 and 16) needs \( O(|E| + |V|^2 \sum_k k_1) \).
- Updating \( U^{(t)} \) (Eqs. 13 and 18) and \( V^{(t)} \) (Eqs. 14 and 18) needs \( O(|V|^2 |E| + |E|^2 |k_1|) \).
- Updating \( R^{(t)} \) (Eqs. 15 and 19) needs \( O(|V|(|k_1 + E|^2)) \).

In summary, the time complexity of our algorithm is follows, where \( \text{iter} \) is the number of outer iterations (lines 3–16 in our algorithm).

\[ O \left( \text{iter} \sum_{t \in T} (|V|^2 k_1 + k_2^2 + |E|^2 |E| + |E| |k_1|) \right) \]  

5 EXPERIMENTAL EVALUATIONS

To demonstrate the applicability and effectiveness of CARNMF, we conducted a set of experiments using real-world datasets. Specifically, the performance of the proposed scheme was compared to simple baseline and state-of-the-art methods.

The experiments were performed on a PC with an Intel Core i7 (3.3 GHz) CPU with 16 GB RAM running Ubuntu14.04. CARNMF was implemented by Python 2.7.6 with Numpy 1.9.0.

5.1 Datasets

We used two datasets: DBLP and arXiv.

- DBLP: Digital Bibliography Project\(^1\) is a bibliographic database in the computer science area. DBLP contains publication information, such as authors and conferences. We used a part of the dataset by extracting conferences similar to [8]. We extracted four research areas: data mining, databases, machine learning, and information retrieval, and five major conferences for each area. Consequently,
Table 1: Selected conferences on four research areas.

<table>
<thead>
<tr>
<th>DB</th>
<th>DM</th>
<th>ML</th>
<th>IR</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIGMOD, VLDB</td>
<td>KDD, ICDM</td>
<td>NIPS, ICML</td>
<td>SIGIR, ECIR</td>
</tr>
<tr>
<td>PODS, EDBT</td>
<td>PKDD, SDM</td>
<td>ECMIL, UAI</td>
<td>JCDL, ECIR</td>
</tr>
<tr>
<td>ICDT</td>
<td>PAKDD</td>
<td>COLT</td>
<td>TREC</td>
</tr>
</tbody>
</table>

Table 2: Selected journals on four research areas.

<table>
<thead>
<tr>
<th>math-ph</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Communications in Mathematical Physics</td>
<td></td>
</tr>
<tr>
<td>Reviews in Mathematical Physics</td>
<td></td>
</tr>
<tr>
<td>Letters in Mathematical Physics</td>
<td></td>
</tr>
<tr>
<td>Journal of Mathematical Physics</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>nucl-ph</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Annual Review of Nuclear and Particle Science</td>
<td></td>
</tr>
<tr>
<td>Progress in Particle and Nuclear Physics</td>
<td></td>
</tr>
<tr>
<td>Atomic Data and Nuclear Data Tables</td>
<td></td>
</tr>
<tr>
<td>Journal of Nuclear Materials</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>astro-ph</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Astronomical Journal</td>
<td></td>
</tr>
<tr>
<td>Annual Review of Astronomy and Astrophysics</td>
<td></td>
</tr>
<tr>
<td>New Astronomy Reviews</td>
<td></td>
</tr>
<tr>
<td>Space Science Review</td>
<td></td>
</tr>
<tr>
<td>cond-mat</td>
<td></td>
</tr>
<tr>
<td>Nature Nanotechnology</td>
<td></td>
</tr>
<tr>
<td>Smart Materials and Structures</td>
<td></td>
</tr>
<tr>
<td>Semi Conductor Science</td>
<td></td>
</tr>
<tr>
<td>Journal of Materials Science</td>
<td></td>
</tr>
</tbody>
</table>

10,491 papers in 20 conferences (shown in Table 1) were selected.

- arXiv: arXiv² is a repository of electronic preprints in various scientific fields. Similar to above, we chose four research areas: mathematical physics (math-ph), nuclear (nucl-th), astrophysics (astro-ph), and materials (part of cond-mat), and four major journals for each area. Consequently, 12,497 papers in 16 journals (shown in Table 2) were selected.

Multi-attributed graphs were constructed from the datasets as follows: The nodes correspond to the authors. If two authors co-author a paper, we placed a weighted edge between the authors. The weighting denotes the number of co-authored papers. Each author has attributes term, paper, and conference/journal, which are defined below:

- term: Each term is regarded as a node. An edge is generated between an author and a term if the author uses the term in the titles of at least one paper. The edge weight denotes the term frequency for each author. As preprocessing, we applied stop-word elimination and stemming.

- paper: Each paper is regarded as a node. An edge is generated if the author publishes the paper. The edge weight is always 1.0 because each paper can only be published once.

- conference/journal: Each conference or journal corresponds with a node. An edge is created between an author and a conference/journal if the author publishes at least one paper at the conference/journal. The edge weight is the total number of publications at the conference/journal.

5.2 Results of CAR-clustering

Figure 1 shows examples of the detected communities and their associated attribute-value clusters in DBLP. The number of communities and the number of term clusters were each 50, whereas the number of conference clusters and the number paper clusters were each 4. The red, blue and gray rectangles correspond to communities, term clusters, and conference clusters, respectively.

Each rectangle shows the top contributing nodes in the community/cluster, and the edge weights show the strength of the relationship between the community and the corresponding cluster. We chose famous researchers in different research domains (i.e., Jiawei Han, Michael Stonebraker, and Michael I. Jordan).

Figure 1(a) show the community and the correlated attribute-value clusters of Jiawei Han, who is a leading researcher in data mining and database areas. The results show that (1) he collaborates with Chinese researchers, (2) he publishes many papers related to data mining and database conferences (i.e., KDD, ICDM, SDM, PKDD, and VLDB), and (3) his researches are highly correlated with topics in data mining, such as clustering and classification on large graph.

Similarly, Figure 1(b) shows the result for Michael Stonebraker, a renowned database researcher. His community is strongly related to conferences in databases (SIGMOD, VLDB, PODS, EDBT, and ICDT). Topics such as management, data metric, and query evaluation are detected. Figure 1(c) shows the result for Michael I. Jordan, an expert in machine learning research. This community is strongly related to the conferences of machine learning, (NIPS, ICML, UAI, COLT, and ECML) and the topics like learning network, expert model, and prediction.

The detected communities and the associated attribute-value clusters seem to be reasonable.

5.3 Accuracy Comparison

The proposed scheme is compared to a baseline method as well as state-of-the-art methods to quantitatively evaluate the performance of community detection and attribute-value clustering. The comparison methods include:

- NMF [15]: Baseline approaches that apply NMF for binary relationships between graph components, including author-term (A-T), author-paper (A-P), author-conference (A-C), term-paper (T-P), and term-conference (T-C)³.

- LCTA [26]: A probabilistic generative model for communities, topics of textual attributes, and their relationships.

- SCI [23]: An NMF based method for detecting communities as well as their semantic descriptions via node’s attribute values.

- HINMF [16]: A model that clusters objects and attributes simultaneously and takes the consensus among the binary NMFs. This work is the most similar to our proposal.

Note that, LCTA and SCI deal with a single concatenated feature of multiple attributes. Therefore, we prepare concatenated feature consisting of term, document and conference/journal, and apply these approaches on the feature.

To evaluate the qualities of these methods, we compared the accuracy [26] w.r.t. community and attribute-value clustering w.r.t. paper and conference/journal. We designed a ground truth to measure the accuracy. To derive the ground truth, each author is labeled based on research areas of their papers, in other words, if the author mostly published papers for the specific area, the author is labeled as that area. Similarly, the labels for conference/journal and paper were manually given by referring to the conference categories.

DEFINITION 5 (ACCURACY). Given a set $S$ of elements, for each element $n \in S$, the true label and the cluster label generated by a method are denoted by $s_n$ and $r_n$, respectively. Then, the accuracy

²https://arxiv.org

³Because NMF assumes the co-occurrences of binary relationships, paper-conference (one-to-one relationship) is excluded.
is defined as:

\[ \text{Accuracy} = \frac{\sum_{n \in S} \delta(s_n, \text{map}(r_n))}{|S|} \]

where \(| \cdot |\) is the cardinality of a set, \(\delta(x, y)\) is a delta function which returns 1 if \(x = y\), otherwise 0; and \(\text{map}(r_n)\) is a mapping function that maps \(r_n\) to the equivalent label in the dataset. The best mapping can be found by Kuhn-Munkres algorithm [13].

Table 4 summarizes the evaluation results. The number of communities and the number of attribute-value clusters for each attribute are each four. Each cell shows the mean and the standard deviation of the accuracies for 20 trials. N/A denotes that the method does not support the category. Values in bold indicate a significant improvement using the Student-t test, where \(p < 0.05\).

CARNMF achieved the best performance for community detection (author) and attribute-value clustering (paper and conference/journal) with significant gaps for DBLP dataset (respectively 11\%, 22\% and 7\%) and for arXiv dataset (respectively 3\%, 4\% and 2\%) relative to the comparative methods. In particular, CARNMF has an improved clustering quality compared to NMF by taking the relationships between communities and attribute-value clusters into account.

Table 5 summarizes evaluation of effects from taking multiple attributes into account. The table showcases results where different combinations of attributes are used, e.g., \("T-C\)" means term and conference attributes were used. This result shows that the proposed method works the best when taking as many attributes as possible. As expected, the basic tendency is that as the number of attributes increases, the accuracy increases.

Table 3 lists the detected topics from DBLP using CARNMF when the number of topics is set to four. Our method successfully detects the four major research topics. Specifically, Topic 1 containing retrieve, inform, search, query and web seems to correspond to information retrieval, and Topic 2 containing mine, pattern, cluster, graph and frequent correspond to data mining. Topic 3 contains words "learn, network, kernel, bayesian, reinforce", which are typical words of machine learning. Topic 4 is a topic of database containing words "query, database, optim, xml, manage", which are popular topics on database researches.

### 5.4 Insights on Parameters

This section discusses the effect of parameter \(\lambda_t\) for each attribute. The larger the \(\lambda_t\) value, the greater the influence of the attribute-value cluster for \(t \in T\) is on the community. Therefore, optimal parameter setting should result in better results. Figures 2 shows the behavior of the accuracy with different values with respect to different attributes. For each evaluation, \(\lambda_t\) \((s \neq t)\) of the other attributes were fixed. In most cases, the accuracy shows a convex form and the peak is around 10\(^{-2}\). More importantly, the accuracy is insensitive to the setting, making tuning easier.

### 5.5 Convergence Analysis

In this section, we experimentally provide convergence analysis to optimize the proposed loss function in Equation 4. Figures 3(a) and (b) show the convergence curve of the loss function for DBLP and arXiv, respectively. In addition, the accuracy of each iteration is plotted. The black line shows the value of the loss function. The red, green, and blue lines show the accuracy of community detection and attribute-value clustering for author, paper, and conference/journal, respectively. As the number of iterations increases, the loss function decreases while the accuracy improves.
Table 4: Quality evaluations of community detection and attribute clustering.

<table>
<thead>
<tr>
<th></th>
<th>DBLP dataset</th>
<th></th>
<th></th>
<th>arXiv dataset</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Author Paper</td>
<td>Conference</td>
<td>Author Paper</td>
<td>Conference</td>
<td>Author Paper</td>
<td>Conference</td>
</tr>
<tr>
<td>NMFA (A-T)</td>
<td>64.02 ± 5.73</td>
<td>N/A</td>
<td>32.75 ± 2.30</td>
<td>N/A</td>
<td>67.81 ± 4.09</td>
<td>N/A</td>
</tr>
<tr>
<td>NMFA (A-P)</td>
<td>43.12 ± 5.17</td>
<td>44.58 ± 5.89</td>
<td>N/A</td>
<td>34.19 ± 3.55</td>
<td>33.53 ± 1.93</td>
<td>N/A</td>
</tr>
<tr>
<td>NMFA (A-C)</td>
<td>75.35 ± 6.85</td>
<td>N/A</td>
<td>87.60 ± 1.73</td>
<td>N/A</td>
<td>67.81 ± 4.09</td>
<td>N/A</td>
</tr>
<tr>
<td>NMF (T-P)</td>
<td>N/A</td>
<td>50.02 ± 7.93</td>
<td>N/A</td>
<td>28.22 ± 9.00</td>
<td>N/A</td>
<td>69.06 ± 1.36</td>
</tr>
<tr>
<td>NMF (T-C)</td>
<td>N/A</td>
<td>N/A</td>
<td>69.88 ± 6.68</td>
<td>N/A</td>
<td>69.06 ± 1.36</td>
<td>N/A</td>
</tr>
<tr>
<td>LCTA</td>
<td>48.90 ± 5.77</td>
<td>26.13 ± 4.36</td>
<td>68.50 ± 12.46</td>
<td>41.08 ± 4.31</td>
<td>33.88 ± 1.82</td>
<td>61.56 ± 9.94</td>
</tr>
<tr>
<td>SCI</td>
<td>54.78 ± 8.79</td>
<td>22.31 ± 1.48</td>
<td>58.20 ± 7.40</td>
<td>38.68 ± 3.56</td>
<td>35.36 ± 0.91</td>
<td>42.81 ± 3.58</td>
</tr>
<tr>
<td>HINMF</td>
<td>68.90 ± 9.08</td>
<td>56.46 ± 3.08</td>
<td>90.10 ± 12.63</td>
<td>65.41 ± 5.38</td>
<td>33.24 ± 2.43</td>
<td>61.25 ± 7.81</td>
</tr>
<tr>
<td>CARNMF</td>
<td>86.34 ± 2.39</td>
<td>78.19 ± 9.87</td>
<td>97.20 ± 5.21</td>
<td>72.65 ± 8.11</td>
<td>42.23 ± 3.18</td>
<td>71.25 ± 2.53</td>
</tr>
</tbody>
</table>

Table 5: Quality evaluations of community detection and attribute clustering, changing attributes to be used.

<table>
<thead>
<tr>
<th></th>
<th>DBLP dataset</th>
<th></th>
<th></th>
<th>arXiv dataset</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Author Paper</td>
<td>Conference</td>
<td>Author Paper</td>
<td>Conference</td>
<td>Author Paper</td>
<td>Conference</td>
</tr>
<tr>
<td>CARNMF (T)</td>
<td>43.29 ± 3.05</td>
<td>N/A</td>
<td>43.19 ± 6.44</td>
<td>N/A</td>
<td>69.06 ± 1.36</td>
<td>N/A</td>
</tr>
<tr>
<td>CARNMF (P)</td>
<td>46.84 ± 4.15</td>
<td>41.69 ± 5.97</td>
<td>33.23 ± 2.63</td>
<td>33.49 ± 1.68</td>
<td>N/A</td>
<td>69.06 ± 1.36</td>
</tr>
<tr>
<td>CARNMF (C)</td>
<td>85.11 ± 2.57</td>
<td>92.40 ± 8.50</td>
<td>67.41 ± 6.85</td>
<td>69.69 ± 6.64</td>
<td>N/A</td>
<td>69.06 ± 1.36</td>
</tr>
<tr>
<td>CARNMF (T-P)</td>
<td>43.28 ± 4.44</td>
<td>41.33 ± 3.88</td>
<td>35.32 ± 2.04</td>
<td>35.79 ± 2.28</td>
<td>N/A</td>
<td>69.06 ± 1.36</td>
</tr>
<tr>
<td>CARNMF (T-C)</td>
<td>83.67 ± 6.54</td>
<td>N/A</td>
<td>95.20 ± 1.99</td>
<td>68.93 ± 7.40</td>
<td>66.88 ± 4.88</td>
<td>N/A</td>
</tr>
<tr>
<td>CARNMF (P-C)</td>
<td>86.41 ± 1.93</td>
<td>73.30 ± 11.98</td>
<td>94.10 ± 3.42</td>
<td>69.71 ± 8.85</td>
<td>40.15 ± 3.13</td>
<td>68.75 ± 5.59</td>
</tr>
<tr>
<td>CARNMF (T-P-C)</td>
<td>86.34 ± 2.39</td>
<td>78.19 ± 9.87</td>
<td>97.20 ± 5.21</td>
<td>72.65 ± 8.11</td>
<td>42.23 ± 3.18</td>
<td>71.25 ± 2.53</td>
</tr>
</tbody>
</table>

Figure 2: Accuracy for different \( \lambda_t \) values.

5.6 Efficiency Analysis

This section analyzes computational efficiency in terms of the numbers of communities and attribute clusters. When the numbers are fixed to four as experiments above, the running times of CARNMF on the DBLP (arXiv) dataset are 1.186 ± 0.253s (0.682 ± 0.138s). When changing the numbers of communities and term clusters to 50, while those of paper and conference remain four, the running times increase to 7.471 ± 0.563s (DBLP) and 6.526 ± 0.172s (arXiv). These values are still reasonable for various applications.

Moreover, we examine the running time of our method by changing the number of nodes in an input graph. Theoretically, as discussed in Section 4.4, the computational complexity is dependent on the number of vertices, that of edges, and that of distinct values of each attribute. As most of real-world graphs are modeled as scale-free networks, edges in a graph are very sparse, therefore, we examine the sensitivity of processing time on the proposed method in terms of the number of nodes. In this experiment, we selected all of the papers on DBLP, and construct the multi-attributed graph as same manner as described in Section 5.1. We set the number of communities and clusters are four.
Figure 3: Convergence analysis of the proposed algorithm to optimize a loss function and the corresponding accuracy curve.

Figure 4: Time complexity of CARNMF w.r.t. the number of input nodes

Figure 4 shows that the time complexity of our method is almost linear to the number of nodes. From the figure, we ensure that the time complexity of our method is linear to the numbers of nodes and edges (as shown on Equation21). Therefore, when the input graph is sparse, our method is highly efficient.

6 CONCLUSION

In this paper we have proposed CAR-clustering, which includes community detection, attribute-value clustering, and extraction of their relationships, for clustering over multi-attributed graphs. We have also proposed a novel algorithm CARNMF based on NMF. CARNMF employs a unified loss function to simultaneously solve the problem. The performances of nodes and edges (as shown on Equation21). Therefore, when the input graph is sparse, our method is highly efficient.

ACKNOWLEDGMENT

This research was partly supported by Japan Agency for Medical Research and Development (AMED).

REFERENCES

[26] Zhijun Yin, Liangliang Cao, Quanquan Gu, and Jiawei Han. 2012. Latent author-topic model for authors and documents. In Proceedings of the sixth ACM international conference on Web search and data mining. ACM, 587–596.
Classifying land cover from satellite images using time series analytics

Patrick Schäfer
Computer Science Department
Humboldt-Universität zu Berlin
Germany
patrick.schaefer@hu-berlin.de

Dirk Pflugmacher
Patrick Hostert
Geography Department
Humboldt-Universität zu Berlin
[dirk.pflugmacher,patrick.hostert]
@geo.hu-berlin.de

Ulf Leser
Computer Science Department
Humboldt-Universität zu Berlin
leser@informatik.hu-berlin.de

ABSTRACT

The Earth’s surface is continuously observed by satellites, leading to large multi-spectral image data sets of increasing spatial resolution and temporal density. One important application of satellite data is the mapping of land cover and land use changes such as urbanization, deforestation, and desertification. This information should be obtained automatically, with high accuracy, and at the pixel level, which implies the need to classify millions of pixels even when only small regions are studied. Balancing runtime and accuracy for this task becomes even more challenging with the recent availability of multiple time points per pixel, created by periodically performed satellite scans. In this paper we describe a novel approach to classify land cover from series of multi-spectral satellite images based on multivariate time series analytics. The main advantage of our method is that it inherently models the periodic changes (seasons, agriculture etc.) underlying many types of land covers and that it is comparably robust to noise. Compared to a classical feature-based classifier, our new method shows a slightly superior overall accuracy, with an increase of up to 20% in accuracy for rare land cover classes, though at the cost of notably increased runtime. The highest accuracy is achieved by combining both approaches.

1 INTRODUCTION

Monitoring changes in land usage is an important area of research as land cover is a key variable driving the Earth’s energy balance, hydrological and carbon cycle, and the provisioning of natural resources and habitat [22]. Over the last three decades, satellite-based Earth Observation (EO) programs have made tremendous progress in acquiring medium-resolution (10 − 100m) images around the globe systematically and with increasing frequency (revisit time). As a result, large volumes of medium-resolution satellite images are now available free of charge, enabling automatic approaches to the identification of land usage and the detection of land surface changes over large areas. For example, the American Landsat 8 sensor images the Earth at 30-m spatial resolution every 16 days, and two European Sentinel-2 sensors acquire images with a revisit time of 5 days and a spatial resolution of 10 − 20 meters, which amounts to roughly 60 measurements for more than 300 Billions pixels (excluding oceans) in a year.

The free availability of medium-resolution satellite image time series has spawned new possibilities for mapping land cover [9]. In the past, classification approaches operated on single images or stacks of images (i.e. composite classification). Multi-date classification approaches exploit the notion that land cover can vary over time, e.g. because of vegetation senescence or harvesting [27]. The task can be approached in different ways. In a typical baseline setting, the different measurements per pixel are used as independent features for a classical machine learning-based classifier, such as Naive Bayes or Decision Trees [20]. In this approach, the temporal order of the measurements is ignored as all features are treated as orthogonal dimensions of the feature space. An alternative method is to include the consideration of the order of measurements by using methods from time series analytics [2, 24]. Here, every pixel is considered as a temporally ordered (and aligned) series of measurements, and the specific changes (increasing or decreasing slopes, periodic changes etc.) of the measurements over time are analyzed to find commonalities and to derive classification models. Previous works (see Section 3) have shown that this can be advantageous as land cover are temporally variable and often follow characteristic temporal patterns, such as those imposed by seasons. However, given the enormous scale of the data to be classified, not only the accuracy of an approach is important, but also the runtime performance has a critical role in any practical application.

In this work, we evaluate the recently proposed multivariate time series classification algorithm WEASEL+MUSE [26] for land cover classification using temporally dense, medium-resolution satellite images. WEASEL+MUSE models multivariate time series using the truncated Fourier transformation and discretizes measurements, both of which to reduce noise, builds a rich feature space to capture also subsequences in the time series, is able to exploit similar temporal subsequences even when appearing at very different offsets within a time series, and uses aggressive feature selection to remove irrelevant features and thereby speedup classification. Although WEASEL+MUSE was not developed specifically for land cover classification, many of its aspects fit nicely to the specificities of this domain, such as the inherent noise reduction and the exploitation of repetitive behaviour.

We compare the prediction performance of WEASEL+MUSE with the performance of an established and popular machine learning-based approach, Random Forests, using the same input features on 23 Landsat 8 images collected in 16 day-intervals over Reunion Island. The study region covers an area of 2866x2633 pixels at 30-m spatial resolution. As reference dataset for model training and validation, we used a sample of 81714 pixels that had been manually classified into 9 land cover classes.

Our results indicate that time-series-based algorithms improve land cover classification accuracy compared to non-temporal algorithms. Our time series algorithm WEASEL+MUSE achieved higher classification accuracies than Random Forests. The improvement in accuracy was most notable with rare and/or difficult classes. Here, class-wise accuracies increased by 8 and 3 percentage points, respectively. Overall accuracy improved by 1%-point owing to the fact that the dominant classes were less affected by
the choice of algorithm. Interestingly, Random Forests captures different signals in the data than the time-series-based approach, as a simple Ensemble of both approaches further improved classification accuracy. However, this increased accuracy comes at the cost of an increased runtime. Thus, we will focus future work on improving the runtime of our method without loosing the advancements in prediction performance we observed.

The rest of this paper is organized as follows: Section 2 provides background on land cover classification using satellite images. Section 3 presents the current state of the art in land cover mapping and time series classification. Section 4 describes the test dataset and the tested classification methods: Random Forests and WEASEL+MUSE. Section 5 presents the experiments.

2 LAND COVER CLASSIFICATION FROM SATELLITE IMAGES

The classification of satellite images to extract information on land cover and land use has a long history. A significant turning point in terrestrial Earth Observation was the launch of Landsat-1 in 1972 (then called Earth Resource Technology Satellite). For the first time, Landsat delivered systematic observations of the Earth land surface for land monitoring, leading to new ways of machine-assisted approaches for mapping land cover from space [7]. In the 1990s and early 2000s, international and national agencies started to adopt operational land cover mapping programs with Landsat and Landsat-like data, e.g. in Europe (CORINE), USA (NLCD), Canada (EOSD), and Australia (NCAS-LCCP). Even decades later, the Landsat program is still active today. Since February 2013, Landsat 8 is taking images of the Earth at 30 m spatial resolution in 8 spectral bands every 16 days. The radiometric quality and spectral resolution has greatly improved since the early satellites, and because of new global acquisition strategies and on-board storage and download capabilities, so has the sheer number of available images. The number of medium spatial resolution (10 – 100 m) sensors has been increasing [4], thus dense time series of medium resolution are available for many parts of the globe.

Multi-spectral sensors like Landsat record the sun’s energy reflected by a surface in a few distinct spectral wavelengths (bands), e.g. blue, green, red in the visible spectrum (400 nm to 700 nm), near infrared (700 to 1100 nm), and short-wave infrared (1100 to 3000 nm). Since land surfaces with different chemical and structural properties often absorb and reflect sunlight differently and wavelength-dependent, information on land cover can be derived from these spectral bands. For example, water absorbs much of the near-infrared radiation, so these wavelengths are useful for discerning land-water boundaries that are not obvious in visible light. Similarly, green vegetation absorbs much of the incoming radiation in the red spectrum while reflecting about 50% of the radiation in the near-infrared spectrum.

Much of the past research on classification algorithms has focused on exploiting the spectral and spatial properties of land covers, including artificial neural networks [1], decision trees [11], support vector machines [15], and spatial segmentation algorithms [10]. Each algorithm has its strength and weakness with respect to: the distributional assumptions made about the data, training requirements, computational complexity, and robustness to overfitting, data noise, and errors in training data. Also common to all algorithms is that the work is supervised, i.e., they need an independent reference data set (i.e., land cover information collected in the field or from air-photos) for training a model. It is fair to say, that no single algorithm works best for all applications and reference data. However, there has been a trend away from parametric statistical models to machine learning to deal with the complexity of input data and class legends.

It has only recently become feasible to build land cover mapping algorithms that exploit the temporal domain of entire pixel time series with medium resolution data [12]. To this end, satellite images typically first undergo a series of pre-processing steps, including the correction of atmospheric effects, geometric alignment and cloud and cloud-shadow masking. Once these steps are finished, spectral values of pixels can be traced over time to identify and detect land surface changes such as deforestation [8], or urbanization [19].

3 RELATED WORK

Although, time-series based classification (TSC) is a relatively new area in remote sensing, the topic itself has a long tradition and dozens of approaches exist (see [2, 24], for instance). Time-series-based classifiers can broadly be categorized into two classes: Similarity-based methods use a similarity measure over sequences, such as Dynamic Time Warping (DTW) [21]. Feature-based TSC method is dynamic time warping (DTW) [21]. Feature-based MTSC can be grouped into those methods that build feature from so-called shapelets [30], which are short and maximally discriminative subsequences of the time series, and methods using the bag-of-patterns (BOP) approach [3]. The standard BOF model [17] break up a time series into windows, represent the corresponding subsequences within each window as discrete features (or words), and finally derives a classifier from the frequencies of the words. Different approaches to TSC (and MTSC) differ not only in their accuracy on different data sets, but also in their runtime for classification, which is a critical issue when it comes to very large data sets as is the case of satellite images. For instance, we recently evaluated the runtime of 12 different state-of-the-art methods for (univariate) TSC and found differences of up to three orders of magnitude [24, 25].

Most approaches to land cover classification rely on traditional machine-learning methods (see previous section), and there have been only a few prior studies on using time series information. For example, [32] fitted harmonic functions to each satellite band and used the fitted parameters as features in subsequent classification (which implies that it falls into the class of feature-based MTSC methods). [14] found that including temporal information into a model can have a bigger impact on classification accuracy than the choice of the particular classification algorithm. New time-series-based algorithms are needed to leverage the predictive potential of satellite time series images [6].

4 METHODS

4.1 Description of Data

For our analysis, we used a public dataset taken from the TiSeLaC (Time Series Land Cover Classification Challenge) [28]. This dataset consists of time series of 23 Landsat 8 images collected in 16 day-intervals over Reunion Island in 2014. Landsat data were provided by the French Pôle Thématique Surfaces Continentales...
Figure 1: Normalized Difference Vegetation Index (NDVI) of 9 different land cover classes for our reference dataset [28].

<table>
<thead>
<tr>
<th>ID</th>
<th>Land cover class</th>
<th>Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Urban Areas</td>
<td>16000</td>
</tr>
<tr>
<td>2</td>
<td>Other built-up surfaces</td>
<td>3236</td>
</tr>
<tr>
<td>3</td>
<td>Forests</td>
<td>16000</td>
</tr>
<tr>
<td>4</td>
<td>Sparse Vegetation</td>
<td>16000</td>
</tr>
<tr>
<td>5</td>
<td>Rocks and bare soil</td>
<td>12942</td>
</tr>
<tr>
<td>6</td>
<td>Grassland</td>
<td>5681</td>
</tr>
<tr>
<td>7</td>
<td>Sugarcane crops</td>
<td>7656</td>
</tr>
<tr>
<td>8</td>
<td>Other crops</td>
<td>1600</td>
</tr>
<tr>
<td>9</td>
<td>Water</td>
<td>2599</td>
</tr>
</tbody>
</table>

Table 1: Land cover classes and distribution of reference samples [28].

In general, a time series dataset contains N time series. Each time series is associated with a class label y from a predefined set of labels Y. Time series classification (TSC) is the task of predicting a class label for a time series whose label is unknown. A classifier is a function that is learned from a set of labelled time series (the training data), takes an unlabelled time series as input and outputs a label. In this paper, each pixel should be labeled by one of the nine reference land cover classes.

The 23 Landsat 8 images contain 10 time series of spectral features each of length 23. At the pixel level, this data can be interpreted as a multivariate time series (MTS).

A multivariate time series (MTS) \( T = \{ t_1, \ldots, t_n \} \) is an ordered sequence of \( n \in \mathbb{N} \) streams \( t_i = (t_{i,1}, \ldots, t_{i,m}) \in \mathbb{R}^m \), i.e., \( m \) recorded values at each fixed time stamp. As we address MTS generated from satellites with a fixed revisit time, we can safely ignore the concrete time stamps. MTS are typically produced by sensors recording data over time like motion captures, gestures, EEG signals, hand-written letters, sign language, or the multi-spectral image data sets captured by satellites.

4.2 Time Series Analytics

Random forests (RF) are an ensemble learning method widely used in Earth Observation (EO) for classifying land cover and land use (e.g. [31]). RF build ensembles of decision trees wherein each tree is trained on randomly selected features of a bootstrapped training sample. Node splits are performed using a random subset of predictor variables. Because of these random components, the RF approach does not require tree pruning and is relatively insensitive to overfitting [5]. To predict a class label, a RF classifies the sample with all its decision trees and returns the mode of the predicted classes. A RF approach to satellite image / pixel classification uses the concatenated spectral features of each pixel as input (this is a vector of length 230 and the 2 coordinates, longitude and latitude). RF are a typical machine learning method which do not consider any order of the features. If time series are used as features, each point in time is considered as an independent feature and the order of measurements is ignored. This implies that such methods are unable to reproduce serial correlations or to detect temporal trends in the data.

4.3 Machine Learning Approach using Random Forests

Random forests (RF) are an ensemble learning method widely used in Earth Observation (EO) for classifying land cover and land use (e.g. [31]). RF build ensembles of decision trees wherein each tree is trained on randomly selected features of a bootstrapped training sample. Node splits are performed using a random subset of predictor variables. Because of these random components, the RF approach does not require tree pruning and is relatively insensitive to overfitting [5]. To predict a class label, a RF classifies the sample with all its decision trees and returns the mode of the predicted classes. A RF approach to satellite image / pixel classification uses the concatenated spectral features of each pixel as input (this is a vector of length 230 and the 2 coordinates, longitude and latitude). RF are a typical machine learning method which do not consider any order of the features. If time series are used as features, each point in time is considered as an independent feature and the order of measurements is ignored. This implies that such methods are unable to reproduce serial correlations or to detect temporal trends in the data.

4.4 Land Cover Classification with WEASEL+MUSE

WEASEL+MUSE (Word Extraction for time SErries cLassification + MIltivariate Symbols and dErivatives) [26] is a state-of-the-art MTS classifier that is composed of the building blocks depicted in Figure 2. It conceptually builds upon the univariate Bag-of-Patterns model applied to each dimension. In the BOP model [23], a time series is characterized by the frequency of occurrence of substructures. BOP-based algorithms build a classification model by (1) extracting subsequences from a time series, (2) transforming each subsequence (of real values, the measurements) into a discrete-valued word (a sequence of symbols over a fixed alphabet), (3) building a feature vector from word counts (histogram), and (4) finally using a classification model from the machine learning repertoire on these feature vectors.

Specifically, WEASEL+MUSE treats the pixel time series of the 10 spectral features as a MTS with 10 dimensions. For each spectral feature, subsequences using varying lengths are extracted,
Figure 2: WEASEL+MUSE Pipeline: Feature extraction, univariate BOP models and the multivariate BOP model.

approximated using the truncated Fourier transform, and discretized into words using equi-depth or equi-frequency binning. A feature vector is built from the words (unigrams) and pairs of words (bigrams) to obtain order awareness. Finally, features are concatenated with the sensor id, to maintain a disjoint word space for each dimension. This high dimensional feature space is subsequently filtered using statistical feature selection (Chi-squared test); finally, a logistic regression classifier is trained, assigning weights to characteristic word in each spectral band.

Because WEASEL+MUSE is multivariate, the algorithm can leverage the multi-spectral information of satellite time series. This is an advantage over univariate time series models that operate on single indices (i.e. vegetation indices), as spectral-temporal patterns may differ from sensor band to sensor band. The feature extraction and selection in WEASEL+MUSE make it interesting for land cover recognition:

• **Features extraction:** The words are derived from subsequences extracted at multiple window lengths in each spectral feature using the truncated Fourier transform and discretization. The Fourier transform reduces noise introduced by preprocessing, such as the cloud mask or geometric alignment, and the different window lengths capture the seasonal trends at different time granularity. Bigrams can capture seasonal trends, e.g., higher intensities in the spectrum in summer than in autumn. By extracting subsequences from the pixel stream, the classifier allows for small shifts in the time line, e.g. a delayed bloom of crops in some regions.

• **Feature selection:** The wide range of words considered also introduces many irrelevant features. Therefore, WEASEL+MUSE applies statistical feature selection to remove irrelevant words from each class. These may be a result of erroneous information introduced by the image capture or preprocessing.

The resulting feature vector is highly discriminative and contains words that are characteristic for each class, which allows the use of fast logistic regression classifier. To perform our analysis, we used the JAVA implementation available from [29].

### 4.5 Ensemble Approach

To understand whether there is value in combining the time-series-based approach and the feature-based approach, we build and tested a third model based on the ensemble of WEASEL+MUSE and RF. Both approaches output class probabilities for each pixel and select the class with the highest probability. Pixels belonging to a unique spectral class may be associated with high class probabilities, whereas pixels with less distinct class membership may have more equally distributed probabilities, e.g., 49% vs 51%. To combine the two sets of class probabilities (2x9 class probabilities), we trained a RF model using the 18 class probabilities from the training dataset as predictors and the corresponding land cover class labels as response.

### 5 EXPERIMENTS

#### 5.1 Experimental setup

**Datasets:** We evaluated our competitors using the described Landsat dataset. The dataset was randomly split into 40857 training and 40857 test samples. The training dataset was used to train each classifier. All reported accuracies are based on the test dataset.

**Competitors:** We performed a series of experiments:

- Build RF on the training dataset and apply it to test dataset.
- Build WEASEL+MUSE on the training dataset and apply it to test dataset.
- Build the combined model on the RF- and MUSE-predicted class probabilities on the training dataset and apply it to the test dataset.

**Training:** For WEASEL+MUSE we performed 10-fold cross-validation on the training dataset to optimize the parameters for the SFA quantization method (equi-depth or equi-frequency binning). To perform the RF analysis, we used the R statistical language and the RF package from [16]. We set the algorithm to build 1000 trees and randomly sampled √p variables as candidates at each split (where p = 232, and p is the total number of predictor variables).

#### 5.2 Results

Table 2 presents the class-wise accuracies and overall accuracy (weighted and simple average F1-scores) on the test samples for each of the three methods. Overall, the combined model had the highest F1-score of 91.1% followed by WEASEL+MUSE with 89.6% and RF with 89.0% (weighted averages). Thus, the choice of the classifier had a relatively small effect on the overall accuracy. However, classes with high F1-scores were also better represented in the training and test samples. The effect on overall accuracy was therefore higher when sample sizes were ignored.

When looking at the results in detail, all classifiers showed high F1-scores of about 90% for all but two classes: "other built-up" and "other crops". For these classes, the accuracy was only ~60% and ~50%, respectively. While RF showed a high precision, WEASEL+MUSE had a higher recall and F1-score, indicating that the temporal profile improved the detection of challenging classes. However, the largest improvement in accuracy was obtained from combining the two classifiers, which further improved the F1-score by up to 20 percentage points for the "other crops" class.
The confusion matrix for the RF (Table 3) gives a detailed picture. "other build-up" is often confused with "urban", and "other crops" is often confused with the "other forests", "urban" or "grassland". This might be a result of an under-representation of these land cover classes in the dataset, as these two classes are the ones with the lowest number of instances (Table 1). On average WEASEL+MUSE required 5.8 ms for a single pixel prediction, as a result of the feature extraction and selection phases prior to classification. The RF took 2.7 ms on average per pixel for classification. For the 7.5 Mio pixels of the study area, this translates into a total, single-CPU runtime of about 5.4 hours for RF compared to 12.2 hours with WEASEL+MUSE. WEASEL+MUSE obtained very promising accuracy for many classes. However we observed a limitation of WEASEL+MUSE and all Bag-of-Pattern approaches when applied in the context of land cover classification, namely the discretization step introduced for noise reduction and to obtain words from real-valued sequences. For discretization, the value range is divided into bins, and each one is associated with a label. However, only a limited number of symbols, typically between 4 to 8, can be used to discretize the value range without negatively impacting accuracy [17]. For the Landsat data the spectral range is between 0 and 1000 and the absolute difference between pixels is important for classification. However, after discretization using 8 symbols (i.e., $a = [1000 - 125]$ and $b = [151 - 250], \ldots$) there can be a difference between 0 up to 125 between the values of the same symbol. This noise reduction is useful for applications like gesture recognition [25], but it seems to be too aggressive here.

### 5.3 Sensitivity Analysis

To better judge the performance of the two classification methods, we tested their sensitivity regarding the weighted F1-score to varying sizes of the training sample (Figure 3). Specifically, it was unclear whether the superior performance of the time series classifier could be replicated with small sample sizes. Starting with a random sample of 10% of the original training sample, we iteratively increased the training sample size up to 100%, and tested all models using the test samples. The results show that WEASEL+MUSE scored consistently higher than RF across all sample sizes. The absolute difference was constant, indicating that both approaches were equally sensitive to the sample size.

### 6 CONCLUSION

Our objective was to test a time-series-based classification approach (WEASEL+MUSE) to earth observation time series for land cover classification and conventional machine learning approach (Random Forests). We reported results of a series of experiments using 23 Landsat 8 images collected in 16 day-intervals over Reunion Island. The reference dataset consisted of a total of 81714 pixels distributed over 9 classes. Our key finding is that the use of temporal information improved the classification accuracy.
Figure 3: Overall accuracy of WEASEL+MUSE and Random Forests (RF) classification with varying train sample sizes, i.e., 10%-100% of the training samples to build the model and 100% to estimate accuracy.

but not for all land cover classes. The improvement in accuracy was highest for rare and difficult classes. For these, a combined classifier was able to improve the F1-score even further. We used Random Forests because they are widely used and robust, but the ensemble would also work with other classifiers that output class probabilities. Regarding classification times, the Random Forests approach was twice as fast as WEASEL+MUSE, as Random Forests do not require feature extraction or selection, as opposed to time-series-based approaches.

Further research is needed to understand how MUSE+WEASEL scales over larger areas, e.g. continental scale. The complexity of land cover processes and their spectral-temporal patterns will probably grow with increasing area size. From an application point of view, our presented method may be of particular interest for mapping agricultural land use patterns. Agricultural land can be highly dynamic and spectrally variable throughout the year [13]. This land use class is therefore likely to benefit from time-series based approaches. Future research could test the performance of our time-series based method for classifying a broader range of crop types and cropping cycles. The presented approaches essentially disregard the spatial neighbourhood of a pixel. Experiments have shown that including spatial information can improve classification accuracies, similar to the results reported in [18].

ACKNOWLEDGEMENTS

This work was partly supported by the German Federal Ministry of Education and Research through the GeoMultiSens project (grant no. 01IS14010B).

REFERENCES


Data Analytics to Improve Co-Operative Education

Shivangi Chopra, Yuheng Jiang, Andrew Toulis, Lukasz Golab
University of Waterloo
Waterloo, Ontario, Canada
{s9chopra,y29jiang,aptoulis,lgolab}@uwaterloo.ca

ABSTRACT
In this paper, we summarize our recent research on applying data analytics to a new application area: co-operative education. Many post-secondary institutions currently offer co-operative programs in which students alternate between on-campus classes and off-campus work terms. We observe that the co-operative process produces a variety of interesting data including job advertisements and performance evaluations. We discuss novel data science methodologies we applied to these datasets and the business insights we obtained.

1 INTRODUCTION
According to the World Association for Cooperative and Work-integrated Education, 275 institutions from 37 countries offer co-operative education (co-op) programs, also known as work-integrated learning. Students enrolled in a co-op program typically alternate between on-campus classes and off-campus work terms/internships. Co-operative education has become popular worldwide as it provides an enhanced learning experience for students and a talent pipeline for employers.

Related Work. Research on co-operative education considers three perspectives: of the student, of the employer and of the educational institution. From the student’s perspective, the focus has been on the impact of co-op on skill and career growth, and on characterizing the attributes that make co-op students successful based on survey data and workplace supervisor evaluations. From the employer’s perspective, there has been work on studying employer expectations and job advertisements attractive to prospective employees. From the institution’s point of view, the focus has been on assessing the effectiveness of and improving co-operative academic programs.

Our Approach. Much of the prior work on co-operative education uses data obtained by surveying students or employers. Since surveys tend to suffer from low response rates, datasets used in prior work contain on the order of 100 datapoints or fewer. We observe that a co-op process at a large university generates a large amount of data that can be collected and analyzed: textual data such as job descriptions, relational data denoting which student applied to/interviewed with which employer, and numeric data such as workplace evaluations. Based on this observation, we have recently collected these datasets and initiated a new research direction on data-driven analysis of co-operative education. In this paper, we present an overview of our research agenda, the datasets and methodologies we have used, our results so far, and directions for future work. We believe that co-operative education is an important new application area that showcases the power of data analytics and data-driven decision making.

We classify our research so far into the following four topics.

1. Job analysis: we perform text mining on job advertisements to understand what types of co-op jobs are available and what skills employers are looking for.
2. Competition analysis: we represent students and employers as graphs, with edges between students who interviewed with the same employer and edges with employers who interviewed the same students. This allows us to find densely-connected subgraphs of jobs and students who compete with each other.
3. Satisfaction analysis: we analyze employers’ evaluations of students’ workterm performance and students’ evaluations of employers to determine whether the participating parties are satisfied with each other. Furthermore, since employers rate students on multiple criteria such as productivity, communication and leadership, we can identify what co-op students are good at and what areas need improvement.
4. Entrepreneurship analysis: we identify co-op jobs created by local startup companies to quantify the effect of entrepreneurship on the co-op market.

Roadmap. The remainder of this paper is organized as follows. Section 2 gives an overview of the co-operative education process and the datasets used in our research. Sections 3 through 6 discuss the four research topics mentioned above. For each topic, we present the motivation, followed by our data-driven methodology and the resulting business insights for students, employers and the institution. Section 7 concludes the paper with directions for future work.

2 PROCESS & DATA OVERVIEW
In traditional post-secondary programs, an academic year is divided into two or three semesters, and students spend some or all semesters on-campus taking classes. In co-operative (co-op) programs, students alternate between on-campus study terms and off-campus work terms, with each work term possibly taking place at a different employer. Thus, in any one semester, some students may be taking classes on campus whereas others may be away on work terms. In order to graduate with a co-op degree, students must take the required number of courses and also compete with each other. Work terms may be one or two semesters long. In a typical post-secondary institution, the undergraduate co-op process takes place every semester for students currently on campus who are seeking a co-op job in the upcoming semester. At the beginning of a semester, employers post job advertisements. Students apply to jobs by uploading their resumes and grade transcripts, and employers interview selected candidates. Finally, hiring decisions are made before the end of the current semester. Then, at the end of the work term (next semester), students and employers evaluate each other.
We have collected ten years of co-op data from a large North American university, having the following schema:

- Student data: student id, academic program
- Employer data: employer id, employer name
- Job data: job id, employer id, semester, location, job title, job advertisement text, salary
- Interview data: student id, job id, academic year of the student at the time of the interview, a binary attribute denoting whether or not the student obtained the job
- Employer evaluations of students: job id, overall numeric evaluation, numeric evaluations on various criteria (communication, problem solving, initiative, etc.)
- Student evaluations of employers: job id, overall numeric evaluation

Our dataset spans from 2006 till 2015 and contains over 138,000 job advertisements, over 37,000 students and over 12,000 employers.

Real datasets usually contain errors and inconsistencies. In our case, the salary field was problematic. Some job postings did not include a salary, perhaps because the salary was negotiable. Some jobs included what appeared to be hourly salary, whereas others specified larger numbers which appeared to be monthly or whole-semester salaries.

In the remainder of the paper, we discuss our analysis of jobs (Sections 3 and 6), interviews (Section 4) and evaluations (Sections 5 and 6).

3 JOB ANALYSIS

3.1 Motivation

We begin with an analysis of job advertisements. We observe that job descriptions are a rich source of information about desired skills, company culture and working environments. Thus, our goal is to extract informative terms from job descriptions: technical skills, soft skills, perks (e.g., free food or proximity to public transit) and other terms indicating the nature of the job. We aim to understand employers’ talent needs and to let students know what types of co-op jobs are available to them. We only use the most recent data (from 2015) for this analysis.

3.2 Methodology

Figure 1 shows an anonymized example of a job description. It includes the following useful information:

- Technical skills: Javascript, Ruby on Rails
- Soft skills: team player, ability to learn
- Job duties: architecting and implementing UI designs
- Desired mindset and attitude: obsessed with technology
- Perks: ping-pong and foosball table, free lunch
- Company culture: casual environment

However, job descriptions are not standardized or well structured, and include administrative and formatting elements such as URLs, contact emails, timestamps, and of course common English words. Our technical challenge, therefore, is to extract useful information from job descriptions.

We address this challenge by designing a parser that extracts job-related attributes from unstructured job descriptions. To remove unnecessary words, we build a vocabulary, call it List A, consisting of publicly available lists of common English words, including the following references:

- Wikipedia: Lists of common misspellings: [1]
- Common abbreviations: [2]
- Top programming languages in IT jobs: [3]
- Top programming languages in Finance jobs: [4]

To summarize the parsing process, each job advertisement (the title and the description) is parsed, words are standardized (i.e., stemmed), and words occurring in List A but not List B are removed. We then group the job advertisements in different ways and identify frequent terms in different groups: jobs obtained by junior vs. senior students, jobs obtained by Engineering vs. Finance students, etc.

3.3 Insights

Below, we give two examples of insight that can be obtained by comparing groups of job descriptions; see [4] for full analysis.

First, we compare jobs obtained by Information Technology (IT) students with those obtained by Finance students. The word clouds with frequently occurring terms in IT and Finance jobs are shown in Figure 2. Soft skills are highlighted in green. We note that soft skills such as communication, teamwork and learning are frequent in both types of jobs; this emphasizes the importance of soft skills in post-secondary curricula. However, hard skills are different: IT jobs mention C++ and Java whereas Finance jobs are more likely to mention MS Excel and accounting. Upon closer inspection, we found that the top five sought-after programming languages in IT jobs are Java (mentioned in 33 percent of job postings), C++ (33 percent), JavaScript (31 percent), C (24 percent) and Python (22 percent). We also found interesting differences in technical skills between IT and Finance jobs.


Figure 1: A sample job description.
in the descriptions of mindsets and work environments: IT jobs are more likely to mention passion, creativity and love (of technology) whereas Finance jobs mention client relationships and interpersonal skills.

Next, we show two Venn diagrams in Figure 3, which characterize the overlap between junior jobs (obtained by lower-year students in years 1 and 2) and senior jobs (obtained by upper-year students in years 3 and 4). Again, IT is on the left and Finance is on the right. All IT and Finance jobs require soft skills such as communication and collaboration. However, junior IT jobs require scripting and HTML whereas senior IT jobs mention advanced technologies: distributed and scalable systems and security. Furthermore, common terms in junior Finance jobs include file, arrange, update and MS Office, which suggests clerical and data entry positions. On the other hand, senior Finance jobs are more likely to mention risk managing, statistics, modelling and investing. These results can help manage the expectations of junior students: it may take until senior years to obtain a co-op position that leverages advanced skills and technologies.

4 COMPETITION ANALYSIS

4.1 Motivation

The previous section discussed job description mining to understand what skills employers are looking for. After advertising jobs, the next step in the co-op process is to select candidates for interviews. In this section, we analyze interview data to determine which groups of students and employers compete with each other. Characterizing the extent of competition is an important business problem. For example, employers may not have a good understanding of the available talent pool and may not be allocating their recruiting resources effectively. Likewise, students may not be aware of the extent of competition for various types of jobs and therefore they may not know which jobs are realistically within their reach. Again, we only use the most recent data for this analysis.

4.2 Methodology

We use a graph mining methodology to characterize competition. We construct two graphs from interview data: a student graph, in which two students are connected if they interview for at
least one job in common, and a job graphs, in which two jobs 
are connected if they interview at least one student in common. 
Next, we run community detection on both graphs using the Lou-
vain Method. The goal of community detection is to cluster the 
nodes in a graph such that nodes belonging to the same clus-
ter/community are strongly connected while nodes in different 
communities are sparsely connected [3].

We illustrate our methodology with a simple example in Ta-
ble 1, drawn from [21], which describes interviews of nine stu-
dents (labelled 1-9) for eight jobs (labelled A-H). Figure 4 shows 
the corresponding student and job graphs. The job graph contains 
two communities, coloured blue and red. We can then colour the 
communities in the student graph based on the job communi-
ties in which the students had the most interviews. For example, 
student community 1, containing students 1–5, is blue because 
these students interviewed for jobs in job community 1 which is 
also blue.

In addition to community detection, we identify nodes with 
high closeness centrality, i.e., nodes with the smallest average 
shortest path length to other nodes. These nodes (jobs) are inter-
esting as they are likely to be multi-disciplinary positions that 
interview a diverse set of students and compete with a diverse 
set of other jobs for these students.

4.3 Insights

Below, we describe selected results on the competition in the 
Information Technology sector; see [21] for full details and see 
[14] for a graph-mining study on the competition for co-op jobs 
among academic programs.

The Louvain Method found eight clusters in our job graphs, 
three of which contained mostly IT jobs. Upon further inspection, 
we established a clear ranking of these three communities:

- The first community contained sought-after IT jobs at 
top companies such as Facebook and Google. Most of the 
students who interviewed for these jobs were senior (in 
their third or fourth years of study).
- The second community contained small IT companies 
and start-ups which mostly interviewed and hired junior 
students (in their second year of study).
- The third community had mostly quality assurance and 
software testing jobs, which are perceived by students as 
less desirable work. Most students competing for these 
jobs were in their first year of study and had little prior 
work experience.

When analyzing competition, we found that some small IT 
companies and start-ups from the second community interviewed 
the same students as top-tier companies from the first community. 
However, a majority of these top students accepted positions from 
top IT companies, and the smaller companies ended up hiring 
more junior students. We conclude that the smaller companies 
that are able to attract significant student attention are underes-
timating their competition and have difficulties competing for 
top co-op talent.

Interestingly, our centrality analysis revealed that the most 
central job in the top-tier IT community was a data scientist 
position, suggesting that data science roles are more multi-
disciplinary than traditional IT positions.

5 SATISFACTION ANALYSIS

5.1 Motivation

Having analyzed what employers are looking for and which 
groups of employers (and students) compete with each other, 
we now turn to analyzing work term evaluations to understand 
whether students and employers are satisfied with each other. 
Additionally, analyzing evaluation sub-categories suggests what 
students are good at and what areas need improvement (as per-
ceived by their co-op employers). This analysis uses the most 
recent three years of data and only includes Engineering students 
(the largest co-op population at the university).
Table 2: Average scores of the 19 sub-categories of performance evaluations of co-op students, in descending order

<table>
<thead>
<tr>
<th>Category</th>
<th>Average score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response to supervision</td>
<td>3.65</td>
</tr>
<tr>
<td>Ability to learn</td>
<td>3.59</td>
</tr>
<tr>
<td>Interpersonal behaviour</td>
<td>3.54</td>
</tr>
<tr>
<td>Dependability</td>
<td>3.53</td>
</tr>
<tr>
<td>Adapting to org. rules &amp; policies</td>
<td>3.52</td>
</tr>
<tr>
<td>Handling conflicts</td>
<td>3.51</td>
</tr>
<tr>
<td>Interest in work</td>
<td>3.45</td>
</tr>
<tr>
<td>Quality of work</td>
<td>3.42</td>
</tr>
<tr>
<td>Quantity of work</td>
<td>3.40</td>
</tr>
<tr>
<td>Integration of prior learning</td>
<td>3.40</td>
</tr>
<tr>
<td>Goal setting</td>
<td>3.31</td>
</tr>
<tr>
<td>Initiative</td>
<td>3.30</td>
</tr>
<tr>
<td>Verbal communication</td>
<td>3.26</td>
</tr>
<tr>
<td>Judgement</td>
<td>3.23</td>
</tr>
<tr>
<td>Written communication</td>
<td>3.22</td>
</tr>
<tr>
<td>Problem solving</td>
<td>3.21</td>
</tr>
<tr>
<td>Planning &amp; organization</td>
<td>3.13</td>
</tr>
<tr>
<td>Creativity</td>
<td>3.01</td>
</tr>
<tr>
<td>Leadership</td>
<td>2.92</td>
</tr>
</tbody>
</table>

5.2 Methodology
The methodology for satisfaction analysis is simple: we compute average evaluation scores for different groups of students and point out statistically significant differences. We also pay attention to the fraction of Not Applicable (N/A) scores as employers have the option to enter N/A for any category that was not applicable to a particular work term.

5.3 Insights
We start with students’ evaluations of their employers (on a scale from one to ten; higher is better). We found that Engineering students gave their employers an average score of 7.55. This suggests that students are generally satisfied with their co-op experience. Interestingly, students tend to rate their first employers higher than subsequent employers, perhaps because their first co-op expectations are lower.

Next, we discuss workplace supervisor evaluations of students. Students receive an overall score from one to five corresponding to: unsatisfactory, satisfactory, good, very good and excellent. We found that Engineering students obtained an average score of 3.74, i.e., between very good and excellent. Senior students consistently obtained higher scores than junior students, and furthermore, senior students were more likely to take a job abroad and be satisfied with it.

Additionally, students are rated on 19 criteria, with each rating being from one to four (higher is better). Table 2 shows the average score for each of the 19 criteria, from highest to lowest. Students tend to excel at Response to supervision and Ability to learn, but are not rated highly on Creativity and Leadership. We speculate that it may be difficult to display leadership and creativity in limited-term co-op positions with well-defined tasks. Students may be focused on completing their tasks before the end of their work term rather than trying out new approaches.

The two criteria with the most N/A scores were Conflict management and Leadership (in fact, nearly half the ratings were N/A). However, the percentage of N/A ratings for Integration of prior learning, Goal setting, Leadership and Written communication decreases from first year through fourth year. This suggests that senior students enjoy more opportunities for leadership and independence. On the other hand, the percentage of N/A ratings for other criteria does not change significantly over time.

Interestingly, we found that the average Problem Solving scores improved the most from first year to final year: they increased from 3.07 to 3.23, which is statistically significant at the 95 percent confidence level.

6 ENTREPRENEURSHIP ANALYSIS
6.1 Motivation
Entrepreneurship can lead to job creation and economic growth. As a result, there has been private and public emphasis on fostering entrepreneurship: examples include tax credits and establishing supporting entities such as startup incubators which are often paired with universities. Furthermore, there is evidence that innovative universities can contribute to growth in the regional economies. Thus, it is natural to ask how entrepreneurship impacts the co-operative process. In this section, we give an overview of our study of the impact of entrepreneurship on co-operative education and job creation [1].

6.2 Methodology
For this analysis, we combine the co-op dataset described earlier with a list of 472 companies started by 746 of the institution’s current or former Engineering students and faculty members. To integrate these two datasets, we matched company and founder names in the startup dataset with employer and student names in the co-op dataset. To deal with alternative name spellings (e.g., “XYZ Inc.” vs. “XYZ Systems Inc.” or “Jim Smith” vs. “James A. Smith”), we identified potential matches using approximate string matching and verified correct matches using publicly available data such as LinkedIn profiles. At the end of this process, we identified:

(1) Co-op placements at the institution’s startup companies, including salaries and students’ and employers’ evaluations
(2) Students in the co-op dataset who at some point were enrolled in a co-op program at the institution and went on to start a company (we refer to these students as future founders)

We then summed up the salaries at the aforementioned co-op placements to quantify the economic impact of entrepreneurship on the institution’s co-op system. For placements with no salary data, we imputed the missing salary with the mean salary across all startup companies.

Since the startup dataset may not be complete, our results should be interpreted as lower bounds on the true number of (and salaries paid by) the institution’s companies. Furthermore, we only consider co-op placements for the institution’s own students, not the total number of jobs created by the institution’s companies.

6.3 Insights
We start with our economic impact analysis. We found that over the past ten years, nearly half (223 of the 472) known companies started by the institution’s Engineering students and professors have participated in the institution’s cooperative process. These
223 companies hired over 5,800 distinct students from the institution, which is 15 percent of all students, for a total of over 9,000 co-op placements, which is 6.5 percent of all placements. We estimate that the salaries paid at these placements add up to over $116 million. The institution can view this as data-supported evidence of the economic impact of the entrepreneurship of its members on co-operative education. Furthermore, these results can be used by institutions to motivate programs and initiatives that encourage entrepreneurship.

We then examined the employer and student evaluations corresponding to placements at these 223 companies. We found that both are statistically significantly higher compared to those at other placements.

Finally, we analyzed the co-op histories of future founders (i.e., students who went on to start companies). We located 221 of the 746 founders in the co-op dataset (the others are faculty or staff members, or students who were not enrolled in a co-op program within the past ten years). Only five percent of these 221 founders are female; in future work, we want to understand why this is the case and to determine if the trend is improving. Notably, future founders were more likely to give and receive higher work term evaluations compared to other students. In particular, future founders were rated more highly than other students for their soft skills such as Initiative, Creativity and Communication (recall Table 2). This suggests a possible link between success in co-operative education and entrepreneurship.

7 CONCLUSIONS

In this paper, we presented a new application area for data analytics: improving co-operative education. We explained the datasets that arise in the co-op process, ranging from textual job advertisements to interview relationships and numeric performance evaluations. We then outlined the data-intensive methodologies that may be applied to produce actionable insight for students, employers and institutions. The methodologies included text mining, graph mining and integrating multiple data sources through approximate string matching.

Our research so far has led to new data-driven insight, but there is more that can be done. Below, we list several potential directions for future work on analyzing co-operative data, possibly combined with other datasets:

- Analysis of co-op and post graduation data: Does co-op employment with a given employer lead to full-time employment with the same employer after graduation? This requires correlating co-op data with postgraduate employment data, which could be obtained, e.g., from LinkedIn profiles.
- Analysis of co-op and secondary school data: Does secondary school work/extracurricular experience help students obtain post-secondary co-op jobs? This requires correlating co-op data with undergraduate admission records.
- Combining competition analysis with satisfaction analysis: do top-tier jobs receive higher evaluations by students?
- Gender equity in co-operative education: Are female students in traditionally male-dominated academic programs such as Computer Science satisfied with the co-op experience?
- Trend analysis: Have sought-after skills changed over time? Have evaluation scores (of students and of employers) changed over time?

- Employer/Employee recommender systems: Can text matching or graph mining techniques such as link prediction be used to recommend potential students to potential employers?

REFERENCES

Recommendation of Job Offers Using Random Forests and Support Vector Machines

Jorge Martinez-Gil
Software Competence Center
Hagenberg GmbH
Hagenberg, Austria
jorge.martinez-gil@scch.at

Bernhard Freudenthaler
Software Competence Center
Hagenberg GmbH
Hagenberg, Austria
bernhard.freudenthaler@scch.at

Thomas Natschläger
Software Competence Center
Hagenberg GmbH
Hagenberg, Austria
thomas.natschlaeger@scch.at

ABSTRACT
The challenge of automatically recommending job offers to appropriate job seekers is a topic that has attracted many research effort during the last times. However, it is generally assumed that there is a need of more user-friendly filtering methods so that the automated recommendation systems might be more widely used. We present here our research on two methods from the data analytics field being able to disseminate job offers to the right person at the right time, which are based on Random Forest and Support Vector Machines respectively. Both methods are used here to identify the actual attributes in which users are set when they are attracted to a job offer. Preliminary results in the context of automatic job recommendation suggest that these two methods seem to be promising.

KEYWORDS
e-recruitment, data analytics, random forests, support vector machines

1 INTRODUCTION
Today, the job market is becoming more and more dynamic. In fact, this is one of the major reasons for an increasing demand for better methods for publishing or finding interesting jobs offers. Moreover, this interest is bidirectional [13], what means that it stems not only from Human Resources (HR) departments in companies, intermediaries or manufacturers of recruiting software, but also from job seekers looking for facing new professional challenges. This means that, as a first step, it is assumed that a preliminary reduction of the most promising applicants and job offers can lead to considerable improvements and savings (in terms of money, time and effort) for both parties [10]. In this context, job portals and online recruitment platforms have been traditionally designed in order to help job providers and job seekers to easily find suitable candidates and job offers respectively.

At present, many job portals and web-based recruitment systems offer their services around the world. However, there is a great corpus of literature suggesting that the functionality of the existing portals could be improved [3, 7, 14–16, 19, 23]. As a general case, only references to online job advertisements are managed, which are then classified using a simple textual description or core attributes. This means that there are serious obstacles for a satisfactory support, at least, in the side of job seekers who are forced to browse through the list of available job offers to find what better fits their needs and interests.

In order to allow job seekers to efficiently find what they are looking for, the research community has been working in a kind of information filtering mechanism (a.k.a. job recommender system [2, 6, 20]) aiming to predict the potential interest of job seekers on given job offers. More specifically, job recommender systems aim automatically suggesting job openings in such a way that as many offers as possible are offered to the right candidates at the right moment.

To appropriate face these problems, a number of alternatives have been already explored: whether data concerning the offer should be provided in a structured or unstructured way [7], which communication channels are the most appropriate in a given context [4, 5], how knowledge extraction over the job descriptions should be performed [22], and so on. However, it is widely assumed that more accurate and user-friendly filtering methods need to be developed in order to reach a wider audience for these kind of software products [18].

Our research work proposes to make this process much more smooth and comfortable for the users looking for accurate job recommendations. In fact, our methods aim to automatically identify the criteria on what potential candidates evaluate the acceptance of a given job offer. Additionally, our research aim to improve the perceived quality of recommendations as feedback is received from users. Therefore, in view of the aforementioned issues, we propose here a novel approach for the accurate recommendation of job offers using two well-known methods from the data analytics field that can have great performance in this context. In fact, the major contributions of this ongoing work can be summarized as follows:

- We propose a novel mechanism to automatically recommend job offers based on Random Forests in an accurate way.
- We propose an alternative mechanism to automatically recommend job offers based on the computation of Support Vector Machines.
- We perform an empirical evaluation of our two proposed methods with real data concerning recruitment from one of our partners.

The remainder of this work is organized in the following way: Section 2 reports the state-of-the-art on existing methods and tools for the automatic recommendation of job offers. Section 3 presents the problem that we are addressing within the frame of this work. Section 4 described our two methods to face that problem, these two methods are based on Random Forests and Support Vector Machines respectively. Section 5 reports the empirical evaluation of our methods. Section 6 outlines the analysis of the results that we have achieved from our empirical evaluation. Finally, we remark the conclusions and the future lines of research.
2 BACKGROUND

For many years, information systems for human resources (a.k.a. Human Resources Management Systems or simply HRMs) have been mainly restricted to tracking applicant’s data through the applicant’s management systems [11]. However, through an increasing differentiation of labor and business worlds, the process of finding the right person for a job opening and vice versa is increasing its complexity. It is clear that upcoming social media channels in addition to an overwhelming number of job portals require new strategies and technologies for both recruiters and job seekers [9].

2.1 Uses Cases

Solutions for the automatic recommendation of job offers are currently of great interest for a number of organizations that wish to automatize their e-recruitment processes. Among the most important ones, we can mention HR departments, market intermediaries, electronic job platforms and portals, or software manufacturers. We offer here a closer overview to each of them.

2.1.1 HR departments. The Human Resources (HR) departments in companies have to daily face with problems of this kind. Currently, the HR departments of large companies receive lots of incoming e-mail applications. All the application documents have to be manually processed, so that the relevant information extracted can be transferred into the internal recruiting systems. This process is very time consuming and spends a lot of resources (time, money, effort). For this reason, only the data from proper candidates should be transferred into the system.

2.1.2 Market intermediaries. HR Recruiters and headhunters usually receive the order of finding the most suitable candidate for a specific job description. The challenge is so complex that many companies are willing to pay big sums for successfully completing this task. Solutions for job recommendation can help to alleviate this problem, so that it can be performed much more efficiently and effectively.

2.1.3 Electronic job platforms and portals. The segment of electronic job platforms and/or portals is subject to a strong competition. To survive in this highly competitive market, these operators provide their customers continually new and additional services. With the envisaged research results in the field of automatic job recommendation, portal operators can increase their level of innovation and therefore generate additional competitive advantages for their customers.

2.1.4 Manufacturers of recruiting software. It is also necessary to mention the manufacturers of recruiting software, since this group is constantly striving to expand their software solution continuously with additional and innovative modules to increase customer satisfaction and generate additional revenue. For this reason, software manufacturers of recruiting solutions are potentially beneficiaries of results leading to a satisfactory job recommendation.

2.2 Existing Recommendation Engines

Existing job portals are mainly based on either the use of relational databases or well-known methods from the area of information retrieval (IR). A major difference between them is that relational systems are only able to work with job offers that are already stored in the databases, while IR-based approaches may allow global searches over the Web or social networks.

When using relational databases, job offers with descriptive attributes such as job title, location, company, required skills, etc. and the URL of the job advertisement are stored in relations, and access is provided by means of database queries in standard languages such as SQL [21]. Consequently, only those vacancies matching exactly the given search criteria can be found [17]. When using IR methods, the full text search is alternatively supported by keywords whereby standard search engines can be integrated. Both procedures can be used in a similar way when searching for offers. However, IR-based methods allow to exploit semantic similarity in keywords, but this is only supported to a limited extent by standard search engines. On the other hand, these approaches generate ordered lists of URLs, where users have a proven tendency to view only the highest ranking results.

For these reasons, and regardless of the way in which job offers are handled and processed, the task of recommending the right offer to the right user has been always an important task [12]. In this way, the research community is working to find ways to make this recommendation fully satisfactory to all parties involved in the process.

2.3 Existing Methods

Techniques for automatic recommendation of job offers are specifically designed to address the problem of information overload by giving priority to information delivery for individual users based on their learned preferences [1].

The most common to process this information nowadays consists of automatically processing the documents involved in the e-recruitment process. For each document, it is possible to extract a vector for each of its fields (which contain textual information) using the bag-of-words model and TF-IDF as weighting function.

Then, some kind of methods for set comparison can shed results on the suitability of a given candidate for a specific job offer.

In general, most of methods try to exploit solutions based on the Vector Space Model (VSM) to measure the similarity ratio between the original job offer and the application received. It is a solution easy to implement, with very low computational costs, and that traditional has achieved very good results in the context of job recommendation. However, new trends bet on the use of machine learning technology in order to overcome the traditional limitations concerning the incapability of going further beyond the syntactical representation of the documents.

3 PROBLEM STATEMENT

The problem that we address within the frame of this work is being able to automatically recommend job offers to the appropriate candidates. We are given past solved cases

$$(x_1, y_1), \quad x \in \mathbb{R}^d, y \in \{-1, 1\}.$$ 

We want a classifier so that,

$$g(x) = \text{sign}(\phi(w) \cdot \phi(x) + b),$$

where

$$\phi(w) \cdot \phi(x) = K(w, x).$$

The key here is being able to evaluate the performance of the proposed method in relation to the past solved cases that are used to feed the algorithm in each iteration to readjust the internal parameters.
4 METHODS

In order to improve the accuracy of the predictions, great research efforts have been made in the last few years concerning the definition of methods for combining a number of simple methods. These methods construct a set of hypotheses (a.k.a. ensemble), and combine the predictions of the ensemble in some way to classify new data. The precision obtained by this combination of hypotheses is usually better than the precision of each individual component. One of the most popular methods in this context are random forests.

On the other hand, algorithms based on n-dimensional geometry where given a set of past solved cases from the past are also gaining popularity. In this way, it is possible to label the classes and train the algorithm to build a geometric model that correctly classify a new sample. We give a deeper insight of these two methods below.

4.1 Random Forests

The first method that we envision in this research work is the Random Forest (RF). The rationale behind RF is to work with a given number of decision trees at the same time. Each tree gives a vote for a given class. This process is iterated by all trees. Then, the RF indicates the results having the most votes.

One of the advantages of RFs using is that, in most situations, this method is able to avoid overfitting of the training set, what it is not always possible by using other machine learning techniques. Figure 1 shows us an example of RF. Please note that, in order to work in a correct way, each decision tree has to been built following these steps:

1. Be $N$ the number of test cases, $M$ is the number of variables in the classifier.
2. The number of input variables to be used to determine the decision on a node is $m$; more $m$ must be always smaller than $M$
3. Select a training set for this tree and use the remaining test cases to calculate the error.
4. For each node of the decision tree, randomly select $m$ variables on which to base the decision. Calculate the best distribution of the training set from the variable $m$.

We think that the main advantages of using RF in this context can be summarized as follows:

- In general, RF has only one parameter to configure, the number of trees in the RF
- Unlike black-box models, the results obtained by RF are easier to interpret
- RF, in general, can be easily extended to support multiple classes
- RF are based on probabilistic principles

4.2 Support Vector Machines

Support Vector Machines (SVM) is a state-of-the-art classification method that separates data samples using the geometric notion of hyperplane. The concept behind SVM is very intuitive and easy to understand: If we have data samples that has been already classified, SVM can be used to generate multiple separation hyperplanes so that the data samples already classified can be divided into segments.

The idea is that each of these segments contain only one class. The SVM technique is generally useful and very accurate in scenarios involving some kind of classification. The reason is that SVM is designed to minimize the classification error and maximize the geometric margin.

From all the classifiers which are able to correctly classify the past samples, we are just interested in picking the closest to the hyperplane. Figure 2 shows us the rationale behind SVM with an example that represents a space of two dimensions. The aim here is to find the hyperplane that best segregates the class of relevant job offers from the class of non relevant job offers. When a new instance is added, then this hyperplane has to be recalculated in order to facilitate future classifications.

SVM has demonstrated a great performance in a number of scenarios involving some kind of classification of data samples in the past. We also think that SVM offers several advantages in the context of automatic recommendation of job offers. These advantages are the following:

- SVM has a regularization mechanism which allows avoiding over-fitting (a.k.a. geometric margin)
- SVM is defined by a optimization problem for which there is a number of existing efficient solutions
- SVM provides an approximation to a bound on the test error, which makes it very robust

SVM also has additional advantage that consists of using kernels, so that it is possible to add expert knowledge about the
Table 1: Average values and standard deviations for the numerical attributes of our data set

<table>
<thead>
<tr>
<th></th>
<th>Average</th>
<th>Std. Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workers</td>
<td>5069.5</td>
<td>9195.2</td>
</tr>
<tr>
<td>Inhabitants</td>
<td>361547.5</td>
<td>642882.9</td>
</tr>
<tr>
<td>Distance</td>
<td>36.3</td>
<td>37.4</td>
</tr>
<tr>
<td>Salary</td>
<td>52437.5</td>
<td>13717.4</td>
</tr>
<tr>
<td>Working time</td>
<td>38.8</td>
<td>3.5</td>
</tr>
</tbody>
</table>

problem. This aspect is out the scope of the present work, but it could be quite interesting to face it as part of our future work.

5 RESULTS

We report here the results from our experiments in the field of automatic job recommendation. We have worked with a data set of 40 job offers that have been evaluated on basis of templates or profiles. A template or profile is a pre-defined pattern that shows interest on job offers that follow certain conditions.

The sample set we are working with is not too large (mainly due to the cost of acquiring data in this context) but it can give us a good starting point to test the accuracy of these methods for solving the problem we are facing.

Before each execution, our complete data set is randomly divided in training set (80% of samples) and test set (20% of samples). The former is intended to train both RF and SVM, and the latter is intended to verify the accuracy of the method.

It is also important to mention that the attributes for each job offers are the following:
- Company name
- Position title
- City
- Distance to home
- Working hours
- Yearly salary before taxes
- Are your potentially interested (Y/N)? (to be predicted)

Table 1 shows us the average values for the attributes and its corresponding standard deviations (the amount of variation or dispersion of the values)

Moreover, the most repeated Position Title is programmer, although other occupations that appear in the data set are analyst, researcher, desk support or developer. The attribute to be predicted is dependent of the profile that we are analyzing. And in some cases it can be strongly unbalanced (what means that it will be an an overwhelming majority samples of one class) what makes the learning process even more difficult. However, this is how things work in real e-recruitment scenarios, where users click in either just a few or in many potential job offers, so we are facing here a realistic situation.

The results will show us the degree of accuracy that we have achieved in each case. In order to identify what is the best strategy in each of these cases, we propose a baseline method that it does not involve any kind of learning.

5.1 Baseline

In order to compare the results from our methods, we need to define a baseline method. Since we want to verify the advantages of using methods being able to analyze past solved cases, we are going to choose a baseline method with no learning capabilities. In this case, we are considering to calculate the average of the attributes for each of the offers that the potential candidate liked in the past. Then, we compare new offers with the ‘average’ one, and we decide if it is similar or not based on the number of similar attributes, i.e. attributes closer to the average.

5.2 Salary driven profile

The first case we are going to study is the profile of a person who is willing to be interested in job offers with very high salaries. Figure 3 shows us the results. Please note that for the RF, we pick the best result since this result can vary depending on the number of decision trees that our method is trying to bag, as we explain later.

It is very important to determine the number of decision trees that we are going to work with. To do that, we run several time the algorithm in order to determine what is the appropriate number of trees to be bagged.

From Figure 4 it is possible to see, the more decision trees we add the better get the results. However, at a certain point the benefit is lower than the cost (in terms of computing time) of including additional decision trees.

5.3 Distance driven profile

In this case, we are going to study the profile of a person who is willing to be interested in job offers for those companies that are located near its current location. Therefore, the template will have Yes in job offers with shorter distances and No in job offers for positions located further away. However, what in principle seems to be an easy scenario, it is not so easy to solve as we...
can see in Figure 5. Reason is that the data set generated by the template is very unbalanced, what means that only a few offers a located in a surrounding area.

In Figure 6, we can see once again how the score improvement decreases as the number of decision trees increases, what means that a larger amount of trees is usually fine just to some extent.

### 5.4 Highly paid hour profile

In this experiment, the template is going to choose those job offers which offers the best hourly rate by the potential employer, i.e. the proportion between salary and work time seems to be more advantageous. This case is quite interesting because it might allow us understanding how our methods behave when the user looks for a complex aggregation of attributes. Figure 7 shows us the results for this experiment.

In Figure 8, we can see once again how, at some point, the improvement of the results decreases as the number of decision trees increases.

### 5.5 Big companies located in big cities profile

In this experiment, the template is going to choose those job offers which are offered by large companies located in big cities. This case is also interesting because it might allow us seeing how our methods deal with the fact that more than one attribute has an impact in the user’s decision. Figure 9 shows us the results of the experiment. As it can be seen, it was not a difficult scenario for any of the methods considered.

For the case of RF, Figure 10 shows us the evolution of the score in relation to the number of decision trees. In this case, the RF remains stable during all the experiments.

### 6 DISCUSSION

From the results that we have achieved in our pool of experiments, it is possible to see that the most important advantages of our approach are:

- Both RF and SVM are quite accurate learning algorithms in the context of automatic job recommendation. For a sufficiently large data set, it is possible to build very accurate classifiers. Even for smaller samples like ours, results are better than those from methods with no learning capabilities.
well, maybe new methods for natural language processing using expert knowledge via kernel neural networks could help in this task. We also would like to an offer is written can help attracting potential candidates as job offers. At that point, we were using just the quantitative methods are quite appropriate for accurately working in the context of large instances. RF and SVM both can handle many variables without discarding any of them, which makes them good candidates to efficiently work at web scale, in large databases or with large instances.

Last, but not least, RF is able to provide useful insights for understanding the interactions between the different variables. On the other hand, SVM operate in a less intuitive way, but in exchange, has had a better performance in most of cases.

However, an complete empirical evaluation over larger data sets should be performed in order to gain deeper insights on the advantages of these two methods. The reason is that, as we have seen, it is not always possible to obtain optimal results with small samples like ours.

7 CONCLUSIONS AND FUTURE WORK

In this work, we have presented our proposal for the automatic recommendation of job offers. Our goal here is being able to build methods being able to deliver appropriate job offers to those job seekers that could be potentially interested on them. To do that, we have based our research efforts on two well-known classification methods: random forests (RF) and support vector machines (SVM).

Our empirical evaluation shows us interesting facts. For example, RF are more likely to be interpreted although they do not present a particularly good performance in relation to SVM. On the other hand, SVM are more accurate, although they work with a model being much harder to interpret by human. What it is clear is, that in both cases, we have shown that these two methods are quite appropriate for accurately working in the context of automatic job recommendation.

As future work, we propose to design novel computational methods being able to process the textual description from the job offers. At that point, we were using just the quantitative information that is advertised. However, we think that the way an offer is written can help attract potential candidates as well, maybe new methods for natural language processing using neural networks could help in this task. We also would like to explore the possibilities to work with expert knowledge via kernel mapping in the case of SVM as we mentioned earlier. Finally, it is also necessary to study how to integrate this technology with existing web information systems so that these two methods can be put into operation by the industry.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their useful suggestions to improve this work. The research reported in this paper has been supported by the Austrian Ministry for Transport, Innovation and Technology, the Federal Ministry of Science, Research and Economy, and the Province of Upper Austria in the frame of the COMET center SCCH.

REFERENCES

[10] Sven Launer, Andreas Eckhardt: Help to find the needle in a haystack: integrating recommender systems in an IT supported staff recruitment system. CPR 2009: 7-12.
[22] Xing Yi, James Allan, W. Bruce Croft: Matching resumes and jobs based on relevance models. SIGIR 2007: 809-810.
Sensitive attribute prediction for social networks users

Younes Abid  
Lorraine University, Nancy, France  
younes.abid@loria.fr

Abdessamad Imine  
Lorraine University, Cnrs, Inria, Nancy, France  
abdessamad.imine@loria.fr

Michaël Rusinowitch  
Lorraine University, Cnrs, Inria, Nancy, France  
michael.rusinowitch@loria.fr

ABSTRACT
Social networks are popular means of data sharing but they are vulnerable to privacy breaches. For instance, relating users with similar profiles an entity can predict personal data with high probability. We present SONSAI a tool to help Facebook users to protect their private information from these inferences. The system samples a subnetwork centered on the user, cleanses the collected public data and predicts user sensitive attribute values by leveraging machine learning techniques. Since SONSAI displays the most relevant attributes exploited by each inference, the user can modify them to prevent undesirable inferences. The tool is designed to perform reasonably with the limited resources of a personal computer, by collecting and processing a relatively small relevant part of network data.

1 INTRODUCTION
Data published on social networks profiles can be mined for inferring sensitive information about users. For instance it was shown in [10] how musical tastes allow one to predict educational level. To increase user awareness about these privacy threats we have designed a tool, SONSAI, for Facebook users to audit their profiles. The system crawls the network around the user and predicts its sensitive attributes values using a machine learning engine. The results provided by SONSAI, also shows the public attributes of the user that have oriented the learning algorithm towards a particular sensitive attribute value. The user can therefore modify these public attributes to prevent inference.

For the approach to be feasible several problems have to be solved: First, data collection by crawling is limited both by the social network and by country regulations. Hence the crawler exploration strategy has to focus only on meaningful representative network nodes. Since attributes are numerous, for the learning program to scale one has to select only the most relevant ones for inferring sensitive attribute values. Hence the second problem is to find an attribute relevance measure that is both accurate and easy to compute. Note that we cannot rely on semantic proximity since we notice that a user that hides a sensitive attribute probably will hide semantically related ones, too. Moreover for fully anonymised datasets the attributes semantics is hard to recover. Therefore we follow an alternative approach by modelling attributes as bipartite graphs and measuring relevance of attributes by comparing their bipartite graph structures.

For specific attributes such as gender and relationship status, the sets of values are much smaller than for other attributes like music and movie. Consequently, the graphs that model these attributes have higher connectivity than the other graphs. For instance, the density of the graph that models gender (as most users publish their gender) is close to 0.5. In order to infer hidden links in such graphs we need to learn from highly connected graphs. However most of the available learning graphs are sparse. To cope with this last problem, we derive new graphs by merging several learning graphs.

All the proposed methods have been implemented for Facebook, however they can be applied to many other social networks. The system has been tested by several volunteer users for auditing their Facebook profiles. In each case a dataset was built from real profiles collected in the user neighborhood network.

Related works. In [15] the authors propose algorithms to detect whether a sensitive attribute value can be inferred from the neighborhood of a target user in a social network. Heatherly et al. [12] infer attribute values in social network with bayesian classification techniques. For the same purpose Estivill-Castro et al. [7] employ decision-tree learning algorithms. In these works learning is performed off-line on large datasets. In order to perform attribute inference from sparser datasets collected in short time by our tool user in his ego-network, we rather use random walk-based learning. The random walk technique has been applied to social representations in [14] and [11] where the authors analysed friendships and used a skip-gram model. In these works, user profiles that have similar friends will be mapped to similar representations. This model helps to detect communities and can predict a set of potential friends for a given user profile. Skip-gram model has also been applied recently to infer social relationship from mobility data [5]. Our work uses a more adapted Continuous Bag of Words (CBOW) model to predict the most likely values for a user sensitive attribute. In our setting friends are considered as an attribute among others. Moreover we first determine automatically a relevant subset of the social network for optimizing random walks. In [3] the relevance of attributes is computed by Bayesian optimisation which is much less efficient than the graph comparison approach adopted here. In particular [3] does not succeed in reasonable time (on standard PC) with attributes like gender and relationship status. Let us note that (unlike [1]) our approach does not need any ontology to perform semantic correlation between attributes. Finally we notice that the proposed system SONSAI is close under some aspects to a recommendation system: an item suggestion can be viewed as an attribute value prediction [4]. However unlike recommendation systems SONSAI also provides explanations for the predicted values, namely an ordered list of attributes that have played a significant role in the computation. For enforcing privacy, our final goal will be

© 2018 Copyright held by the owner/author(s). Published in the Workshop Proceedings of the 21st International Conference on Extending Database Technology (EDBT) (March 26-29, 2018, Vienna, Austria) on CEUR-WS.org (ISSN 1613-0073). Distribution of this paper is permitted under the terms of the Creative Commons license CC-by-nc-nd 4.0.
to reduce the “recommendation accuracy” by acting on this list of attributes.

2 ARCHITECTURE OF SONSAI

The architecture of SONSAI is overviewed in Figure 1. The Facebook crawler (about 5k lines of Java 8) drives a Firefox 58.0b4 navigator through a Selenium 3.5.3 server.

Collected information from each profile, group and page are stored in separated XML files. The Anonymizer, Cleanser, Random walker and Ranker components are written in Python 2.7 (about 2.5k lines of code). The Anonymizer component parses all the XML files, generates anonymized graphs and stores them as TSV files. Then, the Cleanser selects the most relevant graphs and stores them as adjacency lists. The Random walker component browses the adjacency lists and stores the resulting walks in a text document. We use the Python gensim implementation of Word2Vec to parse the text document and compute a vectorial representation of the social network nodes encountered in the walk. Finally, the Ranker component classifies the sensitive nodes according to their similarity to the target user profile.

3 SAMPLING FACEBOOK

We have designed a crawler that explores the social network around a user (to some distance given as a parameter) and collects public information from the visited web pages (i.e. friends, liked pages... ) in order to build a representative subnetwork. We distinguish two types of Facebook nodes: user profiles (u) and pages (p) and two types of links: like-ships between user profiles and pages, and friendships between user profiles. Given a node c, c.n denotes the set of nodes that are linked to c. A discovered node is a node whose URL is known by the crawler. For instance, if the crawler retrieves a user profile and collects its public friends list, then all the friends of that particular user profile are discovered. Algorithm 1 builds at most n_c nodes at distance ≤ d from the target node u_t. Each iteration of the outer loop samples a node, crawls it and updates the sets of discovered and crawled nodes. The sampling is done by random walks of length ≤ d with a transition probability π designed to crawl with higher priority closer nodes and to favour neighbour nodes according to their type. Function sinks(j) returns the set of sinks, i.e. crawled nodes such that all discovered nodes at distance ≤ j are also crawled. Sinks are avoided by the random walks to guarantee that the final node has not been crawled yet.

1 http://www.seleniumhq.org/
2 https://radimrehurek.com/gensim/index.html

4 SOCIAL NETWORK MODEL

Modelling friendship relations. Since friendship on Facebook is symmetric, we model friendship between user profiles by an undirected graph \((U, F)\) where \(U\) is a set of users’ profiles and \(F\) is a set of friendship links between them.

Modelling page like-ships. We model like-ships between user profiles and pages by several bipartite graphs \((U, P, L)\) where \(U\) is a set of users profiles, \(P\) is a set of pages (a type) and \(L\) is a set of like-ships links between them. Figure 2 shows an example of page like-ship modelled by two graphs\(^3\). Graph (a) models liked pages of music type and Graph (b) models liked pages of book type. We note that user profiles can like several pages of the same type.

Anonymizing the social network graphs. For ethical and regulation reasons Facebook identifiers are replaced by fresh identifiers. Each node in the network is then identified by a unique integer ID replacing its Facebook ID. The anonymized IDs are sorted according to the node types. Anonymized graphs are saved under the tab-separated

1 Icon made by Smashicons from www.flaticon.com

2 Icon made by Smashicons from www.flaticon.com
value (TSV) format, one of the most general delimiter-separated values format (DSV). TSV is widely used in graph exchange. In contrast to the dataset released by Netflix\footnote{https://www.kaggle.com/netflix-inc/netflix-prize-data} where only user IDs are anonymised (but not movies title, rating, date of rating), all attribute values are anonymised in our datasets.

In the following a sensitive graph is a graph that models an attribute that is considered sensitive by the user (i.e. the user does not want its value to be predictable). The learning graphs are attribute graphs available for the learning module in order to predict hidden links in the sensitive graph.

5 MODEL CLEANSING

The task of predicting a sensitive attribute from the other ones is made difficult by the size of datasets. Therefore, we suggest to apply the inference process only on a subset of most relevant attributes for the task. Our relevance notion does not rely on semantic proximity since we noticed that i) a user that hides a sensitive attribute will probably hide other semantically-related attributes, and moreover ii) for fully anonymised datasets the attributes semantics is hard to recover. On the contrary we will rely on comparing attributes graph structures.

In the following we assume a fixed sensitive graph $s$.

Step 1: Computing the learning and the confidence rates of learning graphs. In order to compare the structure of a given learning graph to the structure of the sensitive graph, we first split each graph in two parts. The first part contains user profiles that hide their links in the sensitive graph. The ratio of user profiles that publish their links in the first part of the learning graph represents the learning rate $lr$. The second part contains user profiles that publish their links in the sensitive graph. The ratio of user profiles that publish their links in the second part of the learning graph represents the confidence rate $cr$.

The function \texttt{connected_profiles_in()} returns the set of user profiles that publish their links in the graph given as argument. Given $l,s$ respectively a learning graph and a sensitive graph and $U$ the set of user profiles in all graphs, we define:

\begin{align*}
U_l &= \texttt{connected_profiles_in}(l) \\
U_s &= \texttt{connected_profiles_in}(s) \\
lr(l) &= \frac{\text{size}(U_l \cap (U \setminus U_s))}{\text{size}(U \setminus U_s)} \\
hr(l) &= \frac{\text{size}(U_l \cap U_s)}{\text{size}(U_s)}
\end{align*}

Figure 3 depicts an example of splitting two graphs for comparison. The graph that models the link-ship between user profiles and pages of politicians is the sensitive graph. And the graph that models the link-ship between user profiles and pages of musics is the learning graph. The learning rate $lr$ for this example is equal to 50%. And the confidence rate $hr$ is equal to 75%.

Step 2: Computing the distance between a learning graph and the sensitive graph. In this step, we discard user profiles that have a null degree in the learning graph or in the sensitive graph. The Jaccard index between two user nodes $u_1$ and $u_2$ in a given graph $A$ is computed as follows, where the function \texttt{links}$_A(u_j)$ returns the set of nodes to which user node $u_j$ is connected in the graph $A$.

\begin{align*}
J_A(u_1, u_2) &= \frac{\text{links}_A(u_1) \cap \text{links}_A(u_2)}{\text{links}_A(u_1) \cup \text{links}_A(u_2)} \\
J_A(u_1, u_2) &= \frac{\text{size}(U_l \cap (U \setminus U_s))}{\text{size}(U \setminus U_s)} \\
hr(l) &= \frac{\text{size}(U_l \cap U_s)}{\text{size}(U_s)}
\end{align*}

The Jaccard index $J$ between graphs $l$ and $s$ is defined by:

\begin{align*}
H(l) &= \sum_{u_k,u_j \in U_l \cap U_s} \left| J_l(u_k, u_j) - J_s(u_k, u_j) \right|
\end{align*}

In order to compare learning graphs with different sets of common connected profiles $U_l \cap U_s$, we normalize this distance by the maximal Hamming distance that can be obtained on such a set. Hence we define the Hamming rate: $hr(l) = H(l)/M(l)$ where $M(l)$ is

\begin{align*}
M(l) &= \sum_{u_k,u_j \in U_l \cap U_s} \left| \text{Max}(J_l(u_k, u_j), 1 - J_s(u_k, u_j)) \right|
\end{align*}

Step 3: Selecting most relevant graphs for learning sensitive attribute values. We first discard the learning graphs that have a learning rate $hr$ lower than threshold $\theta_{lr}$ since they do not convey enough information. We then discard the graphs that have a confidence rate $hr$ lower than $\theta_{cr}$ since they are considered as unreliable. Finally, from the remaining graphs we only select graphs that have a Hamming rate $hr$ higher than $\theta_{hr}$ since they are the most similar to the sensitive graph.

Densifying graphs

For some sensitive attributes such as gender, age and relationship status, user profiles are linked to at most one value. Moreover the sets of values for these particular attributes are much smaller than for other attributes. Consequently, the graphs that model these attributes are denser than the other ones. In this case, for improving the random-walk based learning process (see Section 6) we need to merge several learning graphs in order to obtain a denser one.

We explain the method with a simple example of gender prediction: we select attribute graphs with high $hr$, $cr$ rates.
but with also a good rate of discrimination between genders, that is the gender of connected users in the graph is unbalanced between male and female. For instance we can select jewelry and fast-food graphs. We merge these graphs by grouping all fast-foods in a unique node and similarly for jewelry as shown in Figure 4 to obtain a new learning graph.

Figure 4: Merging graphs.

6 RANDOM WALK-BASED ATTRIBUTE LEARNING

Representation generation. We plan to translate latent information from the selected graphs (in previous section) to a document. The resulting document holds information about paths in the graphs and their frequencies that can be exploited for inferring proximity of a user node to some sensitive attribute value node. Following [14], paths in the social graph are sampled by random walks. In our case, the walks are executed only in the subset of selected graphs.

Let $G_1, G_2, \ldots, G_n$ be the list of selected learning graphs. Let $U$ be the set of users in all graphs. We assume that the friendship graph is selected and $G_1 = (U, F)$ (otherwise we can simply adapt the computation below). The other graphs are bipartite and we pose $G_i = (U, V_i, L_i)$ for $i > 1$. We introduce quotas to quantify the importance of each graph $G_i$ for inferring secret values of the target sensitive attribute. Each selected graph $G_i$ is assigned a 3-dimensional vector $V_{G_i} = [lr(G_i), cr(G_i), 1 - hr(G_i)]$. The quota of $G_i$ is given by its Mahalanobis distance to the null vector $[0, 0, 0]$. It is computed as follows:

$$q(G_i) = \sqrt{V_{G_i}^T \Sigma^{-1} V_{G_i}}$$

with $\Sigma$ the $3 \times 3$ covariance matrix over the set of selected graph vectors.

To specify the random walk transitions we first define the probability $p_{u,y}$ that being in node $u$ the next node in the random walk is in a selected graph $G_y$:

$$p_{u,y} = \left\{ \begin{array}{ll} \frac{q(G_y)}{\sum_x q(G_x)} & \text{if } deg_y(u) > 0 \\ 0 & \text{otherwise} \end{array} \right. \quad (4)$$

where $deg_y(u)$ is the degree of user $u$ in graph $G_y$. A value node $y$ is followed by a user node chosen uniformly at random from the ones connected to $y$. Assuming the node following user node $u$ is in $G_y$, then it will be chosen uniformly at random from the nodes in $G_y$ that are connected to $u$. Therefore, the transition probabilities are $(G_1$ is the friendship graph):

$$u \rightarrow v : \frac{p_{u,y}/deg_y(u)}{if y \neq 1 and (u, v) \in L_y}$$

$$v \rightarrow u : \frac{1/deg_y(v)}{if y \neq 1 and (u, v) \in L_y}$$

$$u \rightarrow u' : \frac{p_{u,1}/deg_1(u)}{if (u, u') \in F}$$

Figure 5: Example of multi graph random walk.

As illustrated in Figure 5 the document is constructed by connecting all graphs through random jumps between their nodes (see also [14]). At each step the walker state changes and a new word is written in the text document. One step amounts to select a graph where the current node occurs with non null degree, and then to select a node that is connected to the current node in the selected graph. The selected node then becomes the new current node.

In this example we aim to predict liked pages of politicians masked by user profile u3. The sensitive graph is Graph 2 and the learning graphs are Graph 1 and Graph 3. Since the values of the sensitive attributes (the pages of politicians) are labelled (each value belongs to a unique cluster), they are represented by the label of their clusters in the final document. We use a greedy clustering algorithm [3] to define size similar clusters. Pages of politicians that share many common 'likers' end up in the same cluster. For instance the first walk depicted by Figure 5 is $[u_1, u_4, v_{2,3}, u_4]$. But for efficiency the walk $[u_1, u_4, c_{2,2}, u_4]$ is stored instead in the document since the value $v_{2,3}$ belongs to the cluster $c_{2,2}$.

Applying word2vec to compute node representations. We have performed multi-graph random walks in the social network and generated a text document. Walks in the document can be interpreted as sentences, where the words are network nodes. Hence, inferring a link between a user node and an attribute value node is similar to the problem of estimating the likelihood of words co-occurrence in a corpus. We use word2vec [9, 13] to map one-hot encoded vectors that represent words in a high-dimensional vocabulary space to low-dimensional vectors (see [6]). Word2vec
is employed with the Continuous Bag of Words (CBOW) model for predicting a word given its context defined by the \( c = 1 \) words surrounding it \( (c \) is the window size of the context\). Inputs of the model in our case are users and published attribute value. Since there is no order between the attribute values CBOW model is more adequate than Skip-gram. The output of the CBOW model is a vector of size \( v \) representing a probability distribution of co-occurrence between all the words of the context and each word from the vocabulary within a window of size \( c \).

**Ranking sensitive attribute values.** We measure semantic similarity between two nodes by the cosine of the angle formed by the vectors representing the nodes. Cosine similarity is known to take greater account of context information. We rank all the sensitive values by their cosine similarity to the target user profile. The values that have the lowest rank are most likely the sensitive attribute secret values of the target user profile, where secret values are actually the true values of the target user but are not published by him on the social network.

The Figure 6 depicts an example of 2-dimensional vectors that encode 8 nodes: 3 user profiles (\( u_1, u_2 \) and \( u_3 \)), 2 pages of musics (\( v_1, v_2 \)) and 3 clusters of politicians (\( c_1, c_2 \) and \( c_3 \)). The clusters of the pages of politicians are the sensitive values and their vectors are red. The node \( u_1 \) is the target user profile and its vector is blue. The clusters of pages of politicians will be ranked according to their distances to \( u_1 \). And the inference algorithm will infer as most probable pages of politicians to be liked by \( u_1 \), the pages of politicians of the cluster that has the smallest rank (the closest cluster to \( u_1 \)).

In [16] Schakel et al. show that word2vec unsupervised learning algorithm encodes word semantics by affecting vectors in the same direction for co-occurrent words during training. Besides, the magnitude of a vector reflects both the frequency of appearance of related words in the corpus and the homogeneity of contexts. Where a context is a set of words that have high co-occurrence probability in the corpus.

In fact, the words that appear in the same contexts have small angular distances between them. The less overlapping the contexts are, the larger the angular distances between their different words are. However, words that appear in many contexts are represented by vectors that average vectors pointing in many contexts directions. Hence, the vectors magnitude generally decreases with respect to the number of contexts. Moreover, the higher the word frequency is, the higher the chance that it is used in different contexts is. Consequently, the vector magnitude also decreases with respect to frequency. From these remarks, we conclude that the euclidian distance is not a good measure of our inference purpose. Actually, words that appear in many contexts have low magnitude. As a result, their euclidian distances will be small and, using this criteria, they would be considered close even if they do not appear in any common context. For instance, the euclidian distance between the cluster of pages of the most popular politicians will be small even if they are rivals. In the example depicted by Figure 6 the politicians of the clusters \( c_2 \) and \( c_2 \) are rivals. The angular distance between those two clusters is big. However, the euclidian distance is small. Moreover, the euclidean distance between a user that has many friends, for instance the user \( u_1 \) in the Figure 6, and a popular music like "despacito", for instance the page of music \( v_2 \) in the Figure 6, will be small. But popular users do not necessarily like popular musics.

### 7 Experiments

To build datasets we have crawled Facebook profiles of people that live in North-East France and in Île-de-France (Paris). Table 1 gives statistics about the two crawled datasets.

<table>
<thead>
<tr>
<th>Dataset 1 (D1)</th>
<th>$#$ Attributes</th>
<th>$#$ Attribute values</th>
<th>$#$ between user profiles</th>
</tr>
</thead>
<tbody>
<tr>
<td>North-East France</td>
<td>1 929</td>
<td>1 022 860</td>
<td>15 012</td>
</tr>
<tr>
<td>Île-de-France</td>
<td>1 296</td>
<td>298 617</td>
<td>6 550</td>
</tr>
</tbody>
</table>

**Table 1:** Details about the datasets.

### 7.1 Political orientation

From each node \( n \) we perform a random-walk of 800 steps. The dimension of the node representation is taken to be 512. The dimension is usually taken between 100 and 300 for natural languages. However the size of the vocabulary in social networks (equal to the number of nodes) is much higher than in natural languages.

In Dataset 1 the sensitive graph represents the links between 2554 user profiles and 4589 politician pages. For each experiment we generate a new social graph from the dataset by selecting the user profiles that publish their preferences concerning the sensitive attribute (pages of politicians) and at least another attribute. Then we remove all the links in the sensitive graph of 10% of the selected user profiles. The algorithm makes sure that all the nodes in the resulting social graph remain connected. The experiments have consisted then in inferring the hidden links based on information from the learning graphs.

Among the 1928 learning graphs, we selected the ones with learning rate greater than \( \theta_{\text{lr}} = 20\% \), confidence rate greater than \( \theta_{\text{cv}} = 60\% \) and Hamming rate lower than \( \theta_{\text{hr}} = 4\% \).

Table 2 details the 23 selected graphs relevance measures.

<table>
<thead>
<tr>
<th>Attribute graph</th>
<th>$#$ Attributes</th>
<th>$#$ Attribute values</th>
<th>$#$ between user profiles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors/Actresses</td>
<td>23.52</td>
<td>71.73</td>
<td>2.24</td>
</tr>
<tr>
<td>Artists/Directors</td>
<td>23.12</td>
<td>74.54</td>
<td>2.78</td>
</tr>
<tr>
<td>Companies</td>
<td>22.82</td>
<td>73.96</td>
<td>1.69</td>
</tr>
<tr>
<td>Artists</td>
<td>22.68</td>
<td>68.79</td>
<td>2.35</td>
</tr>
<tr>
<td>ApplicationPages</td>
<td>21.68</td>
<td>60.36</td>
<td>3.04</td>
</tr>
<tr>
<td>SportsTeams</td>
<td>21.48</td>
<td>63.93</td>
<td>2.35</td>
</tr>
</tbody>
</table>

**Table 2:** Selected learning graphs in D1 for politicians.
We note that the communities graph has the second greatest learning rate $\text{lr} = 44.97\%$, which means that it holds much latent information about users who hide their likes in the sensitive graph. It also has the maximal confidence rate $\text{cr} = 98.47\%$ and the fifth lowest Hamming rate ($\text{hr} = 1.75\%$) among the 23 selected relevant graphs, which means that its structure is very similar to the structure of the politicians graph. The friendship graph (Users) has the maximal learning rate $\text{lr} = 88.37\%$ and a high confidence rate $\text{cr} = 83.98\%$ since 87.62% of users are connected to this graph. However, this graph has a Hamming rate $\text{hr} = 2.08\%$ greater than the average $\text{hr}$ of selected graphs which means that learned information from this graph is less reliable than learned information from the communities graph.

We use the area under the ROC curve (AUC) as defined in [8] to measure the accuracy of the inferred links. For the defined thresholds ($\theta_{lr} = 20\%$, $\theta_{cr} = 60\%$, $\theta_{hr} = 4\%$) the precision is equal to 0.79. However, the inference accuracy when the 23 relevant graphs are selected randomly is only 0.41. We conducted more tests by selecting manually 3 graphs that are semantically close to politics as follows. Graph $G_1$ models the links between 1246 user profiles and 2357 political organizations, $G_2$ models the links between 1120 user profiles and 1758 political parties and $G_3$ models the links between 39 user profiles and 41 political ideologies. Although the selected graphs seem promising, the inference accuracy is only 0.46. This can be explained by the fact that the selected graphs are very sparse and users are vigilant when publishing their preferences about these attributes. Consequently, the algorithm cannot learn well from them.

We note that the musicians/bands graph was automatically selected by our relevance-based selection method confirming that music and politics are correlated as it was empirically discovered in previous studies [17].

Table 3 summarizes the results of the conducted experiments.

<table>
<thead>
<tr>
<th>Selection based on</th>
<th>accuracy</th>
<th>targets</th>
<th>deleted links</th>
<th>nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relevance (23 graphs)</td>
<td>0.79</td>
<td>252</td>
<td>409</td>
<td>558125</td>
</tr>
<tr>
<td>Random (23 graphs)</td>
<td>0.41</td>
<td>204 (average)</td>
<td>351 (average)</td>
<td>11200 (average)</td>
</tr>
</tbody>
</table>

Table 3: Experimental results

### 7.2 Gender and relationship status

To produce a text document from the social graph, we perform random-walks of 80 steps. Each random walk starts from a different node. The dimension of the node vectorial representation is taken to be 128 since in that case the number of sensitive values is smaller. Moreover learning graphs are obtained by grouping several attribute values in the merging operation (see Section 5). For the experiments, we have generated a new social graph from the dataset by randomly selecting 10% of user profiles that publish the value of their sensitive attributes and we have masked it. Then SONSAI has tried to infer the masked values of the sensitive attribute for each user based on the selected attributes by the cleanser module.

**Relationship status.** The sensitive graph models the relationship status of user profiles. To simplify the presentation we define two meta-relationship status as follows:

- $R_1 = \{\text{Single, Divorced, Separated, Widowed, Complicated}\}$
- $R_2 = \{\text{Domestic partnership, Married, Engaged, Relationship, Civil union, Open relation}\}$

We aim to infer the meta-relationship status of users. Table 4 gives more details about the selected attributes from dataset D2.

We notice that discriminant attributes toward $R_1$ are focused around educations and leisures. On the other hand, discriminant attributes toward $R_2$ are focused around business. The accuracy in AUC of inferring the meta-relationship status is higher than 0.7 in both datasets D1.
and D2 as soon as the target publishes values concerning at least 4 selected attributes by the cleanser.

**Gender.** The sensitive graph models the gender of user profiles. We notice that discriminant attributes toward male are focused around sports, games and software. On the other hand, discriminant attributes toward female are focused around health, home and luxury. The accuracy in AUC of inferring the gender is higher than 0.83 in dataset D1 and higher than 0.67 in dataset D2 as soon as the target publishes values concerning at least 2 selected attributes by the cleanser.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Importance</th>
<th>Discrimination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Education</td>
<td>2.75</td>
<td>88.41 % R1</td>
</tr>
<tr>
<td>Community College</td>
<td>2.74</td>
<td>90.02 % R1</td>
</tr>
<tr>
<td>Consulting Agency</td>
<td>2.71</td>
<td>91.18 % R2</td>
</tr>
<tr>
<td>Sports &amp; Recreation</td>
<td>2.56</td>
<td>91.85 % R2</td>
</tr>
<tr>
<td>Home &amp; Garden Website</td>
<td>2.49</td>
<td>91.86 % R2</td>
</tr>
<tr>
<td>Automotive, Aircraft &amp; Boat</td>
<td>2.48</td>
<td>92.86 % R2</td>
</tr>
<tr>
<td>Locality</td>
<td>2.47</td>
<td>92.59 % R2</td>
</tr>
<tr>
<td>Corporate Office</td>
<td>2.46</td>
<td>91.18 % R2</td>
</tr>
<tr>
<td>News &amp; Media Website</td>
<td>2.42</td>
<td>90.32 % R2</td>
</tr>
<tr>
<td>Financial Service</td>
<td>2.41</td>
<td>90.00 % R2</td>
</tr>
<tr>
<td>Industrial Company</td>
<td>2.40</td>
<td>89.29 % R2</td>
</tr>
<tr>
<td>Educational Consultant</td>
<td>2.02</td>
<td>75.00 % R1</td>
</tr>
<tr>
<td>Playground</td>
<td>1.80</td>
<td>66.67 % R1</td>
</tr>
<tr>
<td>Phone/Tablet</td>
<td>1.70</td>
<td>63.64 % R1</td>
</tr>
<tr>
<td>Plastic Surgeon</td>
<td>1.60</td>
<td>60.00 % R1</td>
</tr>
<tr>
<td>Consulate &amp; Embassy</td>
<td>1.60</td>
<td>60.00 % R1</td>
</tr>
<tr>
<td>School Sports Team</td>
<td>1.53</td>
<td>52.00 % R1</td>
</tr>
<tr>
<td>Dive Bar</td>
<td>1.40</td>
<td>54.55 % R1</td>
</tr>
<tr>
<td>Video</td>
<td>1.44</td>
<td>51.00 % R1</td>
</tr>
<tr>
<td>Playlist</td>
<td>1.41</td>
<td>53.04 % R1</td>
</tr>
</tbody>
</table>

Table 4: Selected attributes in D2 for relationship status.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Importance</th>
<th>Discrimination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sports League</td>
<td>1.12</td>
<td>77.97 % Male</td>
</tr>
<tr>
<td>Recreation &amp; Sports Website</td>
<td>3.80</td>
<td>77.09 % Male</td>
</tr>
<tr>
<td>Video Game</td>
<td>3.90</td>
<td>84.18 % Male</td>
</tr>
<tr>
<td>Cars</td>
<td>3.25</td>
<td>73.15 % Male</td>
</tr>
<tr>
<td>Amateur Sports Team</td>
<td>3.03</td>
<td>72.86 % Male</td>
</tr>
<tr>
<td>Sport</td>
<td>2.80</td>
<td>73.07 % Male</td>
</tr>
<tr>
<td>Jewelry/Watches</td>
<td>2.72</td>
<td>56.82 % Female</td>
</tr>
<tr>
<td>Electronics</td>
<td>2.68</td>
<td>55.16 % Male</td>
</tr>
<tr>
<td>Software</td>
<td>2.52</td>
<td>77.23 % Male</td>
</tr>
<tr>
<td>Outdoor &amp; Sporting Goods</td>
<td>2.35</td>
<td>77.19 % Male</td>
</tr>
<tr>
<td>Women’s Clothing Store</td>
<td>2.35</td>
<td>77.28 % Female</td>
</tr>
<tr>
<td>Home Decor</td>
<td>2.29</td>
<td>54.60 % Female</td>
</tr>
<tr>
<td>Stadium, Arena &amp; Sports Venue</td>
<td>2.28</td>
<td>74.45 % Male</td>
</tr>
<tr>
<td>Baby Goods/Kids Goods</td>
<td>2.14</td>
<td>66.61 % Female</td>
</tr>
<tr>
<td>Kitchen/Cooking</td>
<td>2.08</td>
<td>55.93 % Female</td>
</tr>
<tr>
<td>Bags/Luggage</td>
<td>2.04</td>
<td>55.16 % Female</td>
</tr>
<tr>
<td>Beauty, Cosmetic &amp; Personal Care</td>
<td>2.03</td>
<td>60.59 % Female</td>
</tr>
<tr>
<td>Cosmetics</td>
<td>1.98</td>
<td>60.25 % Female</td>
</tr>
<tr>
<td>Hair Salon</td>
<td>1.92</td>
<td>61.44 % Female</td>
</tr>
<tr>
<td>Home &amp; Garden Website</td>
<td>1.72</td>
<td>55.16 % Female</td>
</tr>
</tbody>
</table>

Table 5: Selected attributes in D1 for gender.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Importance</th>
<th>Discrimination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Playlist</td>
<td>1.41</td>
<td>53.04 % R1</td>
</tr>
</tbody>
</table>

7.3 Processing times

Table 6 displays the processing times. The processor clock is 2.3 GHz. Cleansing and random walk algorithms are not parallelized. Cleansing takes more time than the other processes in the case of gender inference since it handles hundreds of thousands of nodes, compares hundreds of graphs to the sensitive graph and computes their importance. The random walk, in the case of gender inference, is performed on a small graph containing only a few dozen of super-values and a few thousands of user profiles. On the other hand, in the case of politicians inference, the task is performed on larger graphs containing dozens of thousands of values. The machine disposes only of 8GB of RAM memory. Each chunk of 5k steps is stored separately in a text file of about 25MB. Those files are then processed one by one with the computer clock rate because only a small part of them can be compared to the sensitive graph, as few users publish their preferences in both graphs. Hence, their structure is not fairly comparable to the politicians’ graph structure. To cope with this problem we compute a third parameter, the confidence rate cr, that indicates how reliable the structure comparison is.

Table 7 shows that the learning rate parameter lr is important to select the best graphs for inference. However, accuracy does not depend only on this parameter since some graphs such as gender graph that have high learning rate may lead to very low accuracy results.

7.4 Parameter sensitivity analysis

Let us investigate the impact of the cleansing parameters lr, cr and hr. All experiments detailed in this section are conducted on dataset D1 to infer users’ political orientation.

Table 7 shows that only 3 graphs among the 1928 available graphs have a learning rate lr higher than 30%. Based on those graphs, inference accuracy can be very low. For instance, inference accuracy based on gender attribute is only 0.36. Based only on the users (i.e. friendship) graph accuracy is getting better to 0.64. The communities graph gives high accuracy of 0.74 for inferring political views. However, we notice that the best accuracy is obtained when selecting graphs with learning rate between 10% and 40%. Table 7 shows that the learning rate parameter lr is important to select the best graphs for inference. However, accuracy does not depend only on this parameter since some graphs such as gender graph that have high learning rate may lead to very low accuracy results.

Table 8 shows that when the Hamming rate hr decreases, accuracy increases. However, most graphs have a low Hamming rate because only a small part of them can be compared to the sensitive graph, as few users publish their preferences in both graphs. Hence, their structure is not fairly comparable to the politicians’ graph structure. To cope with this problem we compute a third parameter, the confidence rate cr, that indicates how reliable the structure comparison is.

Table 9 shows that the confidence rate, cr, does not give information about the best graph to select when it is considered isolately but it must be coupled with other parameters. For instance, if a given graph g has a high confidence rate but a low Hamming rate, that means that it is a good graph for inference. However, if a given graph g has a high confidence rate and high Hamming distance rate that means that g is probably a bad graph for inferring the sensitive attribute. But a given graph g could be interesting for inference if it has low cr and high hr.

8 CONCLUSION

SONSAI application enables users to predict their sensitive links in social networks from relatively small amount of data and computing resources. Indeed, sensitive data inferences are fast and accurate on typical personal attributes. It should be noted that the friendship graph was not selected among important ones to deduce both the gender and the relationship status of users. This probably means that alternative techniques based solely on homophily would be inaccurate in this context. Moreover, we have observed that the privacy of users is threatened as soon as they start publishing at least three important attributes. These ones are automatically brought to light by SONSAIL, regardless
their semantics and only through a structural analysis of the social network graph. As future work, we plan to incorporate countermeasures into our tool to protect users against attacks that might compromise their privacy. We also plan to enhance SONSAI prediction engine with a tool that permits one to disclose hidden friendship links using adequate combinations of queries provided by the social network [2].

Acknowledgments. This work is supported by MAIF Foundation6.

REFERENCES


6www.fondation-maif.fr/
MLNET - Machine Learning Models for Network Analytics

Pedro Casas  
AIT - Austrian Institute of Technology  
Vienna, Austria  
pedro.casas@ait.ac.at

ABSTRACT
The application of Machine Learning (ML) models to the analysis of network measurement problems has largely increased in the last decade; however, there is still no clear best-practice or silver bullet approach to address these problems in a general context, and only adhoc and very tailored approaches have been evaluated so far. While deep-learning models have provided a major breakthrough in highly-dimensional problems such as image processing, it is difficult to say today which is the best model or most fitted category of models to address the analysis of large volumes of highly-dimensional data collected in operational networks. In this paper we evaluate and benchmark different ML models applied to the analysis of three different and assorted network measurement problems, including detection of network attacks, detection of smartphone-apps anomalies and QoE prediction in cellular networks. We consider an extensive battery of ML models, including both supervised and semi-supervised techniques, as well as ML ensembles such as bagging, boosting and stacking. Proposed models are evaluated using real network measurements coming from operational networks. Results suggest that both neural networks and decision-tree-based models provide in general better results in terms of accuracy and prediction, with a much smaller computation overhead for decision trees as compared to models based on neural networks or support vector machines. In addition, collaborative models, and in particular stacking models, are more robust and perform better than single ML models.

1 INTRODUCTION
Data-driven networking, i.e., the design and management of network systems by the analysis of network measurements, represents a key component for future network management. The high-volume and high-dimensionality of network data provided by current network measurement systems opens the door to the massive application of machine learning approaches to improve data-driven networking problems.

There is however a major challenge in applying machine learning models at large-scale for handling network measurements: selecting the best machine learning model for a specific problem is a complex task - it is commonly accepted that there is no silver bullet for addressing different problems simultaneously. Indeed, even if multiple models could be very well suited to a particular problem, it may be very difficult to find one which performs optimally for different data distributions and statistical mixes.

Deep-learning models are today widely used in multiple signal processing problems, particularly in image processing, where they have shown an outstanding performance. However, neural-networks based models, and particularly deep-learning models, have an inherent problem linked to model visibility and interpretation: a deep-learning model is a black-box which can automatically perform feature selection from input raw data and provide highly accurate predictions, but it is very difficult to understand their functioning. Indeed, it becomes very challenging to understand the reasons of a particular classification result, and in particular to understand the input features leading to such a result, as input features derived directly from a deep neural network architecture can be in general meaningless to a domain expert. This is one of the reasons why their application to networking problems is so far quite limited. In addition, deep-learning models are highly data-eager and training them is extremely costly in terms of computational power, which might term them unsuitable for different networking problems which require periodic re-training or whose labeled data are difficult to get.

In this paper we pose ourselves a simple question: which type of machine learning model should be generally used in the analysis of network measurements? Intuition suggests that rule-based models could be in principle a good match for network analytics, as network protocols are highly structured and operate in a rule basis. We therefore present a comparative analysis of different machine learning models, applied to three specific network analytics problems: the detection of network attacks, the detection of network anomalies and the prediction of network performance in terms of end-use Quality of Experience (QoE).

We consider standard and well known machine learning models, which shall ease the interpretation of results and make them more applicable to common networking practitioners. These include decision-trees - single trees and random forests, naïve bayes models, neural networks, support vector machines and nearest neighbors models. We additionally consider collaborative or ensemble learning models, covering the three basic approaches to ensemble-learning: bagging, boosting and stacking. Rather than finding the best model to explain the data, ensemble learning methods build a set of models and then decide between them with some combinatorial approach, seeking model complementarity. Ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. In principle, if no single model covers the true prediction behind the data, an ensemble can give a better approximation of that oracle, true prediction model. An ensemble of models also exhibits higher robustness with respect to uncertainties in training data, which is highly beneficial for generalization of results. We believe that this study would enable a broader application of machine learning to data-driven networking problems, opening the doors to better and more cost-effective network analytics.

This paper builds on top of our recent early work on ensemble-learning models for network analytics [21], where we explore the application of ensemble-learning techniques to network security and anomaly detection, and on machine learning for QoE prediction [20]. The reminder of the paper is organized as follows. Sec. 2 presents a brief overview on the related work. In
Sec. 3 we briefly describe the evaluated machine learning models. Sec. 4 describes the evaluated network measurement and analytics problems and corresponding datasets. Sec. 5 presents the experimental results of the study, benchmarking the accuracy of the proposed models in the analysis of these problems. Finally, Sec. 6 concludes the paper.

2 STATE OF THE ART

The application of machine learning models to network measurement problems is largely extended in the literature. Traffic prediction and classification are two of the earliest machine learning applications in the networking field. In [2], authors provide a survey on different networking problems which have been addressed by machine learning approaches in the past, including objectives such as traffic prediction, traffic classification, network management, self-configuration, as well as performance analysis and prediction.

There are a couple of extensive surveys and papers on network measurement problems such as network anomaly detection [13, 14] - including machine learning-based approaches [11], machine learning for network traffic classification [18] and network security [15], as well as machine learning models for QoE modeling [19] and prediction [20].

The specific application of ensemble learning approaches is by far more limited. Even if it is generally observed in the practice that ensembles tend to yield better results than single models, only few papers have applied them to problems such as anomaly detection [16] and network security [17]. There is a recent surge on the application of deep learning models for network measurement problems, for example for network anomaly detection [10].

3 MACHINE LEARNING MODELS

In the context of supervised learning there are several approaches for predictive model training based on labeled data. The performance of a particular algorithm or predictor depends on how well it can assimilate the existing information to approximate the oracle predictor, i.e., the ideal optimal predictor defined by the true data distribution. However, knowing a priori which algorithm will be the best suited for a given problem is almost impossible in practice. One could say that each algorithm learns a different set of aspects of reality from the training datasets, and then their respective prediction capability also differs between problems.

In this paper we consider six standard machine learning models previously used in the literature for the analysis of network measurements, including: (i) decision-trees (CART), (ii) Naive Bayes (NB), (iii) Multi-Layer Perceptron (MLP) Neural Networks, (iv) Support Vector Machines (SVM), (v) Random Forest (RF) and (vi) Nearest Neighbors (k-NN). We additionally compare three different approaches to ensemble-learning, including bagging, boosting (AdaBoost) and stacking (Stacking MV and GML). We briefly describe all these approaches next.

3.1 Decision Trees

Classification And Regression Trees (CART) [18] define a classification technique based on a tree graph, where inner nodes correspond to a condition on a feature and leaves are the outcome (i.e., the class). A CART represents a very popular classification algorithm due to its simplicity (it can be easily converted into a rule-based classification system) and readability (it can be graphically represented). The training follows a top-down greedy algorithm that works by iteratively splitting the nodes, using normally an information gain based metric as optimization criterion.

3.2 Naive Bayes

Naive Bayes (NB) is a very simple classifier based on Bayesian statistics [18]. Despite its simplicity, it is widely used as it is very efficient in a number of scenarios, especially in high-dimensional datasets. It works by assuming that features are mutually independent, which is not true in most cases, hence the adjective naive. This assumption allows for an easy calculation of the class-conditional probabilities, using maximum likelihood estimation.

3.3 Neural Networks

Multi-Layer Perceptron (MLP) is an artificial neural network composed of multiple layers of neurons, each of them generally represented by a non-linear function [18]. The layers are fully connected in a feed-forward scheme. Each neuron employs an activation function that maps the weighted inputs to the output that is passed to the following layer. The weights, originally set to random values, are iteratively adjusted during the training phase, using back-propagation.

3.4 Support Vector Machines

Support Vector Machines (SVM) are non-probabilistic binary classifiers [18]. SVM is considered one of the most powerful supervised classification algorithm. It works by representing each feature vector in a multidimensional space and trying to find a linear separation (i.e., a hyperplane) for the classes. In some cases, however, a linear separation of the space is not possible, hence it uses the so-called kernel trick, which implicitly increases the dimensionality of the space, resulting in an easier separation in a much higher dimensional space, due to the increased sparsity.

3.5 Random Forest

Random Forrest (RF) is an ensemble technique based on multiple instances of decision trees, each one based on a different part of the training set, randomly selected. These instances are called bootstrapped samples. The final outcome is generally decided by majority voting among all the bootstrapped samples.

3.6 k Nearest Neighbors

The k-Nearest Neighbors algorithm (k-NN) is a non-parametric approach used for either classification or regression. In both cases, the input consists of the k closest training examples in the feature space. In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors.

3.7 Bagging and Boosting Algorithms

Bagging [6] decreases the variance of the prediction model by generating additional training data from the original dataset. Bagging trains each model in the ensemble using a randomly drawn subset of the training set, and each model in the ensemble is then combined in an equal-weight majority voting scheme. Increasing the training data size using a single input dataset does not improve the prediction accuracy, but narrows the prediction variance by strongly tuning the outcome.
Booster [7] involves incrementally building an ensemble by training each new model instance based on the performance of the previous model. Boosting is a two-steps approach, where one first uses subsets of the original data to produce multiple models, and then boosts their performance by combining them, also using majority voting. Different from bagging, boosting subset creation is not random but depends upon the performance of the previous models, and every new subsets contain the misclassified instances by previous models.

We take decision-tree based models for both bagging and boosting, which is a very common approach. In the case of bagging, we consider a Bagging Tree model. We take an AdaBoost [8] Tree model for boosting, which uses decision trees as first level learners. AdaBoost (short for Adaptive Boosting) trains subsequent models in favor of those instances misclassified by previous ones. AdaBoost is sensitive to noisy data and outliers, but in general, it can be less susceptible to over-fitting.

### 3.8 Stacking

While bagging and boosting generally use the same type of model in all the different training steps (e.g., decision trees), stacking [9] aims at exploring the input data space through base models of different type. Stacking is the ensemble learning model that really makes use of a meta learner, which uses the output of the base learners as input for prediction. The point of stacking is to explore a space through the different properties of different models, each of them capable to learn some part of the problem, but not the whole space. The meta learner is said to be stacked on the top of the other based models, hence the name.

General ensemble learning approaches might be prone to overfitting the data. In [1] a simple stacking learning algorithm named **Super Learner** is proposed as a possible solution for this overfitting limitation. It proposes a method to minimize the overfitting likelihood using a variant of cross-validation. In addition, the Super Learner provides performance bounds, as it performs asymptotically as good as the best available single hypothesis predictor, for each predicted pattern.

In the study, we consider two flavors of Super Learner for stacking, using the aforementioned single models as base learners: a simple majority voting based algorithm (Stacking MV), where the output of the base learners are equally weighted to decide on the final output, and GML (Generic Machine Learning), which basically computes weights in an exponential fashion, using the classification accuracy of each base learner. This approach permits to reduce the influence of low accuracy base predictors.

### 4 SCENARIOS AND DATASETS

In this section we overview the three network measurement problems we consider for evaluation. These include: (i) detection of network attacks [3], (ii) detection of smartphone-apps anomalies [4] and (iii) QoE prediction in cellular networks [20].

#### 4.1 Detection of Network Attacks

The first problem consists of the analysis of diverse types of network attacks in real network traffic measurements collected at the WIDE backbone network, using the well-known MAWILab dataset for attacks labeling [12]. MAWILab is a public collection of 15-minute network traffic traces captured every day on a backbone link between Japan and the US since 2001. Building on this repository, the MAWILab project uses a combination of four traditional anomaly detectors (PCA, KL, Hough, and Gamma, see [12]) to partially label the collected traffic.

The traffic studied in this paper spans two months of packet traces collected in late 2015. From the labeled anomalies and attacks, we focus on a specific group which are detected simultaneously by the four MAWILab detectors, using in particular those events which are labeled as "anomalous" by MAWILab. We consider in particular 5 types of attacks/anomalies: (1) DDoS attacks (DDoS), (2) HTTP flasherrows (mptp-la), (3) Flooding attacks (Ping flood), and two different flavors of distributed network scans (netscan) using (4) UDP and (5) TCP-ACK probing traffic. We train ML models to detect each of these attack types separately, thus each detection approach consists of five different detectors which run in parallel on top of the data, each of them specialized in detecting one of the five aforementioned attacks types. As a result, each detection approach can not only detect the occurrence of an attack, but also classify its nature.

To detect different attacks, we consider a slotted, time-based evaluation. For doing so, we split the traffic traces in consecutive time slots of one second each, and compute a set of features describing the traffic in each of these slots. In addition, each slot is assigned a label \( l_t \) consisting of a binary vector \( l_t \in \mathbb{R}^{5 	imes 1} \) which indicates at each position if anomaly of type \( j = 1..5 \) is present or

<table>
<thead>
<tr>
<th>Field</th>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tot. volume</td>
<td># pkts</td>
<td>num. packets</td>
</tr>
<tr>
<td></td>
<td># bytes</td>
<td>num. bytes</td>
</tr>
<tr>
<td>PKT size</td>
<td>pkt_{[min,avg,max,\text{std}]}</td>
<td>min/max/std, PCT</td>
</tr>
<tr>
<td></td>
<td>pkt_{p[1,2,\cdots,95,97,99]}</td>
<td>H(\text{PCT})</td>
</tr>
<tr>
<td>IP Proto</td>
<td># ip protocols</td>
<td>num. diff. IP protocols</td>
</tr>
<tr>
<td></td>
<td>ip_{h}</td>
<td>H(\text{IP})</td>
</tr>
<tr>
<td></td>
<td>ip_{p[1,2,\cdots,95,97,99]}</td>
<td>min/max/std, IP</td>
</tr>
<tr>
<td></td>
<td>ip_{p[1,2,\cdots,95,97,99]}</td>
<td>H(\text{IP})</td>
</tr>
<tr>
<td></td>
<td># icmp/tcp/udp</td>
<td>share of IP protocols</td>
</tr>
<tr>
<td></td>
<td>pkt_h</td>
<td>H(\text{PCT})</td>
</tr>
<tr>
<td></td>
<td>pkt_{p[1,2,\cdots,95,97,99]}</td>
<td>H(\text{PCT})</td>
</tr>
<tr>
<td>IP TTL</td>
<td>ttl_{[min,avg,max,\text{std}]}</td>
<td>min/max/std, TTL</td>
</tr>
<tr>
<td></td>
<td>ttl_{p[1,2,\cdots,95,97,99]}</td>
<td>H(\text{PCT})</td>
</tr>
<tr>
<td></td>
<td># ip_{src/dst}</td>
<td>num. unique IPs</td>
</tr>
<tr>
<td></td>
<td>top_{ip_{src/dst}}</td>
<td>most used IPs</td>
</tr>
<tr>
<td></td>
<td>top_{port_{src/dst}}</td>
<td>most used ports</td>
</tr>
<tr>
<td>TCP/UDP ports</td>
<td>port_h</td>
<td>H(\text{PORT})</td>
</tr>
<tr>
<td></td>
<td>port_{p[1,2,\cdots,95,97,99]}</td>
<td>H(\text{PORT})</td>
</tr>
<tr>
<td>TCP flags (byte)</td>
<td>flags_h</td>
<td>H(TCPF)</td>
</tr>
<tr>
<td></td>
<td>flags_{p[1,2,\cdots,95,97,99]}</td>
<td>H(TCPF)</td>
</tr>
<tr>
<td></td>
<td>% SYN/ACK/PUSH/_</td>
<td>share of TCP flags</td>
</tr>
<tr>
<td>TCP WIN size</td>
<td>win_h</td>
<td>H(WIN)</td>
</tr>
<tr>
<td></td>
<td>win_{p[1,2,\cdots,95,97,99]}</td>
<td>H(WIN)</td>
</tr>
</tbody>
</table>
### Table 2: Input features for anomaly detection.

<table>
<thead>
<tr>
<th>Field</th>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS_query</td>
<td>querycnt</td>
<td># DNS requests</td>
</tr>
<tr>
<td></td>
<td>apn_h</td>
<td>H(APN)</td>
</tr>
<tr>
<td></td>
<td>apn_avg</td>
<td>APN</td>
</tr>
<tr>
<td></td>
<td>apn_p[99,75,50,25,05]</td>
<td>percentiles</td>
</tr>
<tr>
<td>APN</td>
<td>error_code_h</td>
<td>H(Error_flag)</td>
</tr>
<tr>
<td></td>
<td>error_code_avg</td>
<td>Error_flag percentages</td>
</tr>
<tr>
<td></td>
<td>error_code_p[99,75,50,25,05]</td>
<td>percentiles</td>
</tr>
<tr>
<td>Manufacturer</td>
<td>manufacturer_h</td>
<td>H(Manufacturer)</td>
</tr>
<tr>
<td></td>
<td>manufacturer_avg</td>
<td>Manufacturer</td>
</tr>
<tr>
<td></td>
<td>manufacturer_p[99,75,50,25,05]</td>
<td>percentiles</td>
</tr>
<tr>
<td></td>
<td>os_h</td>
<td>H(OS)</td>
</tr>
<tr>
<td></td>
<td>os_avg</td>
<td>OS</td>
</tr>
<tr>
<td></td>
<td>os_p[99,75,50,25,05]</td>
<td>percentiles</td>
</tr>
<tr>
<td>OS</td>
<td>req_fqdn_h</td>
<td>H(FQDN)</td>
</tr>
<tr>
<td></td>
<td>req_fqdn_avg</td>
<td>FQDN</td>
</tr>
<tr>
<td></td>
<td>req_fqdn_p[99,75,50,25,05]</td>
<td>percentiles</td>
</tr>
<tr>
<td>FQDN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

not in current time slot. We compute a large number of features describing a time slot, using traditional packet measurements including traffic throughput, packet sizes, IP addresses and ports, transport protocols, flags, etc. Tab. 2 describes the set of n=245 features, which are computed for every time slot i = 1..n. Note that besides using traditional features such as min/avg/max values of some of the input measurements, we also consider the empirical distribution of some of them, sampling the empirical distribution at many different percentiles. This provides as input much richer information, as the complete distribution is taken into account. We also compute the empirical entropy $H(i)$ of these distributions, reflecting the dispersion of the samples in the corresponding time slot.

### 4.2 Detection of Apps Anomalies

In [4] we conceived a semi-synthetic dataset for traffic anomalies in cellular networks by using real DNS traffic measurements. After collecting DNS traces for longer than six months in 2014 at a cellular network of a large-scale European operator, we devised a technique to generate new traffic traces by carefully recombining real traffic traces. Basically, we take samples of manually labeled one-minute intervals from the original data, characterized by a vector of features containing the distribution of DNS query counts by device Manufacturer, device OS, APN, domain name (FQDN) and DNS transaction flag. With the anomaly-free intervals we generate new synthetic background traffic, simply by shuffling the data samples of the same time of the day and same day class (working or festivity). Then, three different types of anomalies are introduced into the synthetic data, derived from real anomalies observed in this operational network. These anomalies mimick different types of service outages, and are represented by impacting a different number of end-users requesting particular services on specific domain names. The different anomalies considered are E1: short lived (hours) high intensity anomalies (e.g., 10% of devices repeating a request every few seconds), where the involved devices share the same manufacturer and OS; E2: several days lasting low intensity anomalies (e.g., 2% of devices repeating requests every few minutes) and E3: short lived variable intensity anomalies affecting all devices of a specific APN. The used dataset consists of a full month of synthetically generated measurements, reported with a time granularity of 10 minutes time bins. Each time bin is assigned a class, either normal (label 0) or anomalous (label 1, 2 or 3 for the three anomaly types respectively). The dataset includes 16 different variations of E1, E2 and E3 anomalies, impacting a different fraction of end-users - going from 0.5% to 20%. Full details on the synthetic dataset are available in [4].

We take as main traffic feature the total number of DNS requests issued within a time bin. As we saw in [4], perturbations in this feature indicate that a device sub-population deviates from the usual DNS traffic patterns, thus pointing to potential anomalies. To better detect and diagnose the anomalies, we additionally take the distributions of DNS query counts across the aforementioned fields (Manufacturer, OS, APN, FQDN and DNS flag). From these distributions, we compute a set of features describing their shape and carried information, such as various percentiles and entropy values. Tab. 2 describes the specific set of n=36 features, which are computed for every time bin. The set includes the number of observed DNS requests, as well as multiple percentiles of fields such as associated APN, device OS and manufacturer, requested FQDN and number of DNS error messages. We also take as input the average values of these fields, as well as their entropy, the latter reflecting the dispersion of the observed samples.

### 4.3 Cellular QoE Prediction

For the sake of QoE prediction in cellular traffic, we use network and QoE measurements collected in a user field trial taking place in 2015 and detailed in [20], where 30 users equipped with their own devices connected to their preferred cellular operators evaluated three apps as part of their normal daily Internet activity during two weeks: YouTube (watching short videos); Facebook (timeline and photo-album browsing); and Gmaps (satellite maps browsing). QoE feedback was reported for each session through a customized QoE crowdsourcing app, according to a discrete, 5-levels ACR Mean Opinion Score (MOS) scale, ranging from "bad" (i.e., MOS = 1) to “excellent” (i.e., MOS = 5). In addition, each device has a passive flow-level traffic monitor which
records flow-level network traffic statistics, associating flows to apps generating them. The 10 different session-based KPIs in Tab. 3 are derived from the flow-based measurements, which are then synchronized to the QoE feedbacks (MOS scores) using time stamps. The KPIs include features such as average and maximum flow throughput per session, flow size, duration, average signal strength, RAT, ISP, locations, etc. The prediction problem consists in predicting the correct MOS score value (5-classes classification problem), using the session-based KPIs as input. Full details on the dataset are available in [20].

5 EVALUATION AND DISCUSSION

We now evaluate and compare the performance achieved by the presented ML models. To limit biased results, presented results correspond to 10-fold cross validation. All models arc tested using scikit-learn python implementations. Parameters on each different algorithm are calibrated based on standard grid-based search tests. In addition, classes in each classification problem are balanced by statistical bootstrapping [20] to avoid unbalanced training issues. We start by comparing the performance achieved by the machine learning models, and then present a full comparison including also the ensemble-learning approaches.

5.1 Single Base Learning Models

We start the analysis for the case of network attacks detection. Detection performance is measured by computing the True and False Positive Rates (TPR/FPR) for each model and for each of the attack types, using as input the full set of 245 features. Fig. 1 depicts the Receiver Operating Characteristic (ROC) curves obtained with each model, for the proposed attack classes. Besides the ND and the k-NN models, the tested approaches provide all highly accurate results for the five types of attacks. In general, detection performance is worse for DDoS attacks for all the evaluated models, suggesting that its fingerprint in the considered set of features is less marked than for the other attacks. Both the MLP and the RF models achieve the best performance, detecting around 80% of the attacks without false alarms.

Figs. 2 and 3 report the detection (prediction) performance achieved by the six base learning models for the case of anomaly detection and QoE prediction respectively. In both cases, detection is done along with classification (i.e., multi-class problems), thus we study the performance of each detector considering all the classes together. Performance is measured in terms of global classification accuracy (i.e., correctly classified instances), as well as per class recall and precision. Similar to the network attacks problem, Fig. 2 shows that both MLP and RF models are the most accurate ones for the sake of anomaly detection, even if some of the anomalies are more challenging to be correctly detected, for all the models. Indeed, anomalies of type E2 are harder to detect, basically due to their long-lasting, low intensity nature. In terms of QoE prediction, Fig. 3 clearly shows that decision-tree based models, and in particular RF ones, represent by far the most accurate approach, for all the different quality levels.

We focus now on the specific performance of the RF model for the three studied problems. To dig deeper into the RF outperformance, Fig. 4 depicts the ROC curves obtained with the RF model. We include in Fig. 4(a) the performance achieved by the RF model in the detection of network attacks for the sake of comparison to the other two problems. The first observation one could draw is that the RF model is much better for the anomaly detection and QoE prediction problems, but has a comparatively quite poor performance for detection of network attacks, i.e., when comparing at the problems/use-cases level. This point out to the first hypothesis we did in the introduction, suggesting that it is challenging to find a single model to properly tackle different problems simultaneously.

In the specific case of anomaly detection, Fig. 4(b) confirms that while anomalies of type E1 and E3 are perfectly detected by the RF model, anomalies of type E2 are quite often mis-classified as normal operation traffic. Fig. 4(c) shows that the RF model is
very accurate to correctly spot out bad quality sessions (i.e., MOS = 1, 2 and 3), but is less accurate to correctly predict higher quality ones. To better understand these mis-classification issues, Fig. 5 reports the corresponding confusion matrices obtained with the RF model in both problems. As reported in Fig. 5(a), about one third of the E2 anomalies go completely undetected within the normal traffic. In addition, as depicted in Fig. 5(b), excellent quality sessions (i.e., MOS = 5) are often misclassified as good ones (MOS = 4), and as average ones (i.e., MOS = 3) to a lesser extent.

**5.2 Including Ensemble Learning Models**

To conclude with the study, we now include the ensemble learning models within the analysis. Tabs. 4, 5 and 6 report the results obtained with all the models in the three problems, using the Area Under the ROC Curve (AUC) as performance metric. The machine learning community most often uses the ROC AUC statistic for model comparison [5], which is simple and informative.

Tab. 4 depicts the results obtained in the detection of the network attacks. The performance achieved by the ensemble learning models is generally higher than that of the single base learners alone. However, given that the single models performance is
already quite high, improvements are not that much significant. Boosting and bagging provide similar performance for all types of attacks, but it is stacking, and in particular the GML model, which provide the best results. Indeed, GML achieves the highest performance for all the five considered attack categories. In the case of anomaly detection, as previously observed in Fig. 2, almost every predictor achieves an AUC over 99% for E3 anomalies. Thus, there is little room for improvement, which leads to only very subtle differences between the performances of base and ensemble-learners. Still, stacking learning models tend to outperform both first level learners, as well as the bagging and boosting trees. Similar observations can be drawn from the detection of E3 anomalies. Note that the GML model systematically achieves the best results. For E2 anomalies, not only all predictors performed relatively poor, but also many of them achieve very low performance; e.g. the bagging models achieve an AUC below 90%, clearly worse than any other ensemble technique. This scenario highlights the advantages of the stacking models, and in particular, the GML model.

Finally, in the case of QoE prediction, tab. 6 reveals again that predicting excellent QoE sessions (i.e., MOS = 5) is more challenging than for the rest of the quality levels. The CART and random forest models alone provide already very good results, as also shown in Fig. 3. Still, similar to the anomaly detection scenario, the GML model is capable to boost the prediction of excellent QoE w.r.t. both CART and random forest by at least 5% to 10%, suggesting again a better fit for this scenario.

While it is true that in some cases there is not enough room for improvement with respect to bagging, boosting and base learning models, the GML stacking model systematically outperforms these models for most of the tests and in all the problems, which opens the door for generalization of a technique for network measurement analysis. Indeed, stacking is less widely used than
bagging and boosting, but has recently shown outstanding performance in model competitions such as the Netflix Prize [22] and Kaggle competitions (https://www.kaggle.com/). The performance increase as compared to other ensemble learners is small, but in this case the computational overhead is similar, thus the GML model looks more appealing.

6 CONCLUDING REMARKS

In this paper we have demonstrated the outstanding performance of neural networks and decision trees for the analysis of network measurements coming from multiple and assorted networking problems. Based on the performance benchmarking, we can observe that both neural networks and decision-tree-based models provide in general better results in terms of accuracy and prediction than other single models, with a much smaller computational overhead for decision trees as compared to models based on neural networks or support vector machines. Decision-tree based models represent therefore a very appealing machine learning model for network analytics, not only because of their high accuracy and low computational cost, but also due to a series of embedded properties, such as model visibility, robustness to input noise, etc.

When looking at more complex models based on ensembles, we have shown the advantages of ensemble learning techniques to improve detection and prediction accuracy, and particularly of the stacking GML approach. We found that not only the GML based model has the ability to perform as well as the best input single base level learner, but often achieve better results. This includes also the case of both bagging and boosting models, which are also generally outperformed by the stacking models. Performance improvements are higher in scenarios where the performance of the base predictors is relatively low; when first learners performance is already high, there is little room for improvement. We believe that this study would enable a broader application of machine learning models to network analytics problems, with very promising results.

ACKNOWLEDGMENTS

The research leading to these results has been funded by the Vienna Science and Technology Fund (WWTF) through project ICT15-129, "Big-DAMA". Visit https://bigdama.aist.ac.at/ for details on the Big-DAMA project: Big Data Analytics for Network Traffic Monitoring and Analysis.

REFERENCES

Unsupervised Video Semantic Partitioning Using IBM Watson
And Topic Modelling
Alexandre Quemy
IBM Software Lab
Cracow, Poland
Faculty of Computing, Poznań
University of Technology
Poznań, Poland
aquemy@pl.ibm.com

Krzysztof Jamrog
IBM Software Lab
Cracow, Poland
krzysztof.jamrog1@pl.ibm.com

Marcin Janiszewski
IBM Software Lab
Cracow, Poland
marcin.janiszewski@pl.ibm.com

ABSTRACT
In this paper, we present the problem of Video Semantic Partitioning that consists in breaking a video into semantically independent blocks. Defined within a framework of optimization, we present a preliminary heuristic approach to solve the problem, called Split-and-Merge. The algorithm itself is unsupervised, but the mechanisms to extract data from videos are supervised (for some) since they used IBM Watson Services. Finally, we demonstrate on few videos the capabilities of our prototype and discuss the limitations and future improvements. From the experiments, we draw two conclusions: (i) the optimal solution to the problem varies from human to human with a large variability from video to video, (ii) Split-and-Merge demonstrates encouraging qualitative results to find the average optimal solution defined as the average solution given by humans.

1 INTRODUCTION
The problem of text segmentation, that is to say partitioning a text into semantic blocks, has been widely studied (e.g. in [2, 5]) but, as far as we know, never extended to videos. Indeed, video segmentation usually refers to the process of extracting objects from a video and not breaking it into semantic chunks. In this paper, we define the Video Semantic Partitioning problem as an extension of the text segmentation problem and present our primary attempt to solve it. Among the possible applications, we can highlight a better understanding of the video content for practical algorithms and thus we assume we merge the similar cutting points of \( \pi_k \) at any time if necessary.

For a video, the primary source is not as well defined: is it the raw information is totally unstructured and has to be extracted, contrary to text segmentation where this information is structured (the text is already provided as it is). By raw, we mean the primary source of semantic. For a given text, it mainly holds in the sentences and words themselves, while the support brings some secondary elements (e.g. in [2] the support is a HTML page and the tags are used to delimit some sections). For a video, the primary source is not as well defined: is it the audio track or the visual information? Most likely a combination of both. Additionally, this raw information is not directly suitable for most algorithms and requires a pre-treatment to extract exploitable data. Those techniques are known as Cognitive Computing, i.e. processing and analyzing large unstructured signals.

The information extraction plays a preponderant role in the algorithms performances because even the best possible algorithm would return poor results on badly extracted information which is equivalent to noise.

The plan of this paper is as follows: Section 2 formalizes the Video Semantic Partitioning problem, Section 3 presents the feature extraction mechanisms while in Section 4 the algorithm Split-and-Merge is presented. In Section 5 we discuss an optimized version of the algorithm under a restrictive assumption about the videos. In Section 6, we present some results obtained on real videos and conclude this paper in Section 7.

2 VIDEO SEMANTIC PARTITIONING
A video is a temporal signal on a time interval \([0, T]\). We denote by \( P([0, T], m) \) the set of partitions of \([0, T] \) into \( m \) elements, i.e. the elements of \( P([0, T], m) \) are of the form \( \pi = (p_0, ..., p_m, p_{m+1}) \), \( \forall i \in \{0, ..., m\} \), \( p_i < p_{i+1} \), with \( p_0 = 0 \) and \( p_{m+1} = T \), such that \( \bigcup_{i=0}^{m} [p_{i}, p_{i+1}) = [0, T] \).

Our goal is to find a partition such that each \([p_i, p_{i+1})\) is an independent semantic block, i.e. holds a different semantic than its adjacent blocks. Let \( \pi^* \) denotes the optimal partition. The first formulation of the Semantic Partitioning problem is to find a minimizing sequence \( \pi_k \) such that \( \pi_k \rightarrow \pi^* \) with a convergence notion that implies that the number of cutting points of \( \pi_k \) converges to those of \( \pi^* \) and that each cutting point of \( \pi^* \) is the limit of a cutting point of \( \pi_k \).

As an element of any partition \( \pi \) is contained in a closed interval defined by two cutting points of \( \pi^* \) we can define the following cost function \( \forall \pi, \forall \pi^* \in P([0, T], m), \forall p_i \in \pi, c(p_i) = |p_j - p_i|/|p_j - p_{j+1}| \) with \( p_i \in [p_j, p_{j+1}) \). Additionally, we define

\[
J(\pi, \pi^*) = \sum_{i=1}^{m-1} c(p_i) + c(p_{i+1}).
\]

Under the assumption that \( \pi \) has as many cutting points as \( \pi^* \), then if \( J(p, p^*) = 0 \) then \( p = p^* \) and the problem can be seen as finding a minimizing sequence \( \pi_k \) s.t. \( J(\pi_k, \pi^*) \rightarrow 0 \). In practice, as \( \pi^* \) is unknown, there are two problems: the optimal number of cutting points is not known (i) and \( J \) is not available (ii).

For (i), assuming \( J \) is available, if \( \pi_0 \) contains enough cutting points and some in each interval induced by \( \pi^* \), we can minimize \( J \) given \( \pi_0 \) by removing the unnecessary cuttings points. Assuming we know how to remove the unnecessary cutting points, if \( \pi_{n_k} \) is defined by the uniform partition s.t. \( \forall p_i, p_{i+1} \in \pi_{n_k}, p_{i+1} - p_i = h \), then \( \forall h_1, h_2 \) s.t. \( h_1 < h_2, J(\pi_{n_k}, \pi') < J(\pi_{n_k}, \pi^* \}

44

\textsuperscript{1}We may not accept the case such that two cutting points are identical, creating an empty interval, thus with an empty semantic. However, it may be convenient for practical algorithms and thus we assume we merge the similar cutting points of \( \pi_k \) at any time if necessary.

© 2018 Copyright held by the owner/author(s). Published in the Workshop Proceedings of the EDBT/ICDT 2018 Joint Conference (March 26, 2018, Vienna, Austria) on CEUR-WS.org (ISSN 1613-0073). Distribution of this paper is permitted under the terms of the Creative Commons license CC-by-nc-nd 4.0.
For (ii), we need to find an approximation of $J$. As $p^*$ is a partition into semantic blocks we will try to rely on the semantic contained in the video to approximate $J$. To do so, we are able to identify a features stream from the video, that is to say, $n$ sequences of features on $[0,T]$. For each $i \in \{1,...,n\}$, we have a similarity metric $d^i : [0,T] \times [0,T] \rightarrow [0,1]$ to quantify how related two blocks are w.r.t. a particular feature stream. In particular, $\forall p_1, p_2, d_i([p_1, p_2], [p_1, p_2]) = 1$ because the semantic of a block is obviously equal to its own semantic. However, $[p_1, p_2] \cap [p_1, p_2] = 0$ does not imply $d_i([p_1, p_2], [p_1, p_2]) = 0$, i.e. it is not because two blocks are not overlapping in time that they do not have semantic similarity. To ease the notations, given a partition $\pi$, we denote by $d^i_\pi$ the similarity between the blocks $[p_j, p_i]$ and $[p_j, p_i+1]$, i.e. $d^i_\pi = d_i([p_j, p_i], [p_j, p_i+1])$.

The problem of the video partitioning is to find a partition $\pi \in \bigcup_{k \in \mathbb{N}} P([0,T], m)$ that minimizes the adjacent block similarity, that is to say

$$\pi^* = \arg \min_{\pi \in \bigcup_{k \in \mathbb{N}} P([0,T], m)} \left( \sum_{1 \leq j < n} \sum_{j=1}^m d^j_\pi \right)$$

with $\oplus$ an aggregation operator.

For the purpose of this paper, we will consider $\oplus$ as a convex combination of the block similarity, allowing us to reformulate the problem as follows:

$$\pi^* = \arg \min_{\pi \in \bigcup_{k \in \mathbb{N}} P([0,T], m)} \left( \sum_{1 \leq j \leq n} \sum_{j=1}^m \omega_j d^j_\pi \right)$$

(2)

The choice of the operator $\oplus$ is an interesting problem by itself because it is known to greatly influence the algorithms capacity to find the optimal solution [7]. In the future, we plan to investigate the choice of this operator, and in particular treating the problem as multi-objective (optimizing on each feature stream independently) using the Hypervolume [1] or $\varepsilon$-dominance [4] metric as aggregation function.

As said before, one major difficulty comes from the fact the optimal number of elements in the partition is not known a priori. Notice that in practice, if an interval is too small the information it holds is considered as null. For instance, a very small interval does not hold more than few words or no word at all, no screenshot, etc. As a result, its similarity with another very small interval would be one or close to one. In other words, dividing too many times $[0,T]$ does not minimize the objective function. However, having two large blocks separated by a very small block would be a minimizing strategy: the small block holds no semantic, contrary to the large blocks and as a result, the similarity between adjacent blocks is (close to) zero. To handle this problem, we see several possibilities: taking into account this phenomenon into the definition of the metrics $d_i$ (i), adding a regularization term on the block sizes to (2) to be sure they are homogeneous enough (ii), fixing the set of possible values for $m$ and a distribution of block size w.r.t. T (iii). As (i) is too ad-hoc and we do not see any practical justification to support (ii), we will make a stronger assumption on the optimal partition in Section 5 that will define both the maximal number of blocks and their respective sizes.

### 3 FEATURE STREAMS

In this section, we review the feature streams we extract from a video as well as the associated block similarity.

We are able to extract a frame at any moment of the video. For two frames, we are able to calculate two metrics: a perceptual hash $\text{dh}_{\text{phash}}$ using phash open-source library2 and a pixel-based hamming distance $d_{\text{pixel}}$. For two blocks $[p_j, p_i]$ and $[p_j, p_i+1]$, we made the choice to take into account resp. only the last and the first frame contain in the first and second block. With Watson Visual Recognition and for each frame, we are able to obtain a list of tags with confidence on concepts and entities. For a block $[p_j, p_i+1]$, we aggregate the tags of all frames it contains into a list of tags with $\text{confidence}$. To ease the notations, given a block $[p_j, p_i+1]$, we aggregate the tags of all frames it contains into a list of tags with confidence on concepts and entities. For a block $[p_j, p_i+1]$, we aggregate the tags of all frames it contains into a list of tags with confidence on concepts and entities.

**Example:** Let us assume $W_j = \left( \frac{w_1}{w_2}, \frac{w_3}{w_4} \right)$ composed of three words, associated to the confidence vector $C_j = \left( \frac{c_1_j}{c_2_j}, \frac{c_3_j}{c_4_j} \right)$. Similarly, let us assume $W_{j+1} = \left( \frac{w_1}{w_2}, \frac{w_3}{w_4} \right)$ with $C_{j+1} = \left( \frac{c_1_{j+1}}{c_2_{j+1}}, \frac{c_3_{j+1}}{c_4_{j+1}} \right)$. Then, $W_{j,j+1} = \left( \frac{w_1}{w_3} \right)$ and the left and right confidence vectors are given by $C_{j,j+1} = \left( \frac{c_1_j}{c_3_j}, \frac{c_2_j}{c_4_j} \right)$ and $C_{j+1,j+1} = \left( \frac{c_1_{j+1}}{c_3_{j+1}}, \frac{c_2_{j+1}}{c_4_{j+1}} \right)$.

If $|W_j \cap W_{j+1}| = 0$, the similarity is zero: $d_{\text{tags}}^j = 0$. Otherwise, the similarity measure is defined as the Jaccard index ponderated by the confidence:

$$d_{\text{tags}}^j = \frac{|W_j \cap W_{j+1}|}{|W_j \cup W_{j+1}|} \left( 1 - \frac{|C_{j,j+1}^L - C_{j,j+1}^R|}{|W_j \cap W_{j+1}|} \right)$$

$$= \frac{|W_j \cap W_{j+1}|}{|W_j \cup W_{j+1}|} - \frac{|C_{j,j+1}^L - C_{j,j+1}^R|}{|W_j \cap W_{j+1}|}$$

where $|A|$ denotes the cardinal of $A$. In other words, two blocks are similar not only if they have the same tags but also if their confidences are close enough.

**Example:** Considering the vectors defined in the previous example, the similarity between the two blocks is

$$d_{\text{tags}}^{j+1} = \frac{2 - [(c_1_j - c_1_{j+1}) + (c_3_j - c_3_{j+1})]}{4}$$

In this preliminary work, we did not consider OCR to extract the text from the frames or image segmentation techniques to obtain the location of objects on a frame. However, using Watson Speech-to-Text, we extract the transcript from the audio track. The service provides the timestamp to be able to locate the text corresponding to a given block. Similarly to Watson Visual Recognition, Watson Natural Language Understanding is able to extract from a text a list of keywords and concepts with a confidence level (expressing the relevance among the text). This allows us to define $d_{\text{keywords}}$ and $d_{\text{concepts}}$ in a similar way as for $d_{\text{tags}}$.

We normalized the transcript using NLTK [6] which includes the tokenization, removing the stopwords, the lemmatization,
the computation of n-grams from 1 to 4 and filtering the n-grams by a minimal number of occurrences. A given partition \( \pi \in P[0, T], m \) can be seen as a corpus of \( m \) documents represented by Bag-of-Words. Using Gensim [8], we applied a TF-IDF transformation on each block and used a Latent Dirichlet Allocation [3] to express each block in a latent topic space. LDA is a generative probabilistic model that represents documents as a mixture of topics. Each topic represents a (sparse) distribution over the dictionary of words used in the documents expressing the idea that a topic is defined by a high frequency of few terms. The model parameters are then estimated from the real data, in particular the word distribution per topic and the topic distribution for each document. For each pair of adjacent blocks, we use the LDA model to determine the distance between the blocks denoted by \( d_{\text{topical}}^i \). LDA has three hyperparameters: \( K \) the number of topics, \( \alpha \) the document-topic prior distribution and \( \eta \) the topic-word priori distribution. The last two parameters control the sparsity of the distribution. In this work, we used the online hyperparameter strategy provided by Gensim, i.e. the values for \( \alpha \) and \( \eta \) are deduced from the real data. As LDA can be seen as a dimensionality reduction method, the number of dimensions \( K \) influences the quality of the model, thus should be fixed not too low to keep enough information but not too high for the documents to be summarized. For this paper, knowing the approximate size of the Bag-of-Words representation of our data, we fixed \( K = 10 \). Further work should focus on hyperparameter tuning taking into account the length of the video and the number of cutting points at a given moment in a partition (because it influences the transcript length of a block).

We would like to draw the attention on the fact that if the algorithm we used is not supervised, the feature extraction is supervised for some feature streams. The transcript obtained with Watson Speech-to-Text is rectified (including abbreviation and specific term annotations) and inserted into a custom model. As a result, from video to video, the feature extraction becomes more and more relevant, which we believe is the primary requirement to achieve a good semantic partitioning. Similarly, Watson Visual Recognition service is trained to recognize certain objects in relation with the theme of the test videos (e.g. trained to recognize specific softwares displayed in the video).

4 SPLIT-AND-MERGE ALGORITHM

The algorithm is composed of two phases. In the first one, called Split, we break the video into regular blocks that are small enough to be sure they are smaller than the smallest semantic block of the video and thus, in particular, there are several cutting points within each of the intervals defined by the optimal partition \( p^* \). The Merge phase consists in merging two adjacent blocks if they are related enough according to the metrics we defined in the previous Section. Iteratively, we merge the blocks with the higher normalized combined distance defined by \( F(\pi) = \frac{1}{n|\pi|} \sum_{i \leq j \leq m} \sum_{l=1}^{\pi} \omega_l d_i^l = \frac{1}{n|\pi|} \sum_{i \leq j \leq m} F_i^j(\pi) \) with \( F^j_i(\pi) = \sum_{l=1}^{\pi} \omega_l d_i^l \).

However, without a stopping criterion all the blocks will be merged. To prevent this, we take into consideration the improvement implied by removing \( p_j \) between step \( k \) and \( k + 1 \):

\[
R(\pi_k, j) = \frac{F(\pi_k)}{F(\pi_{k+1})} \quad \text{with} \quad \pi_{k+1} = \pi_k \setminus \{p_j\}.
\]

Split-and-Merge is detailed in Algorithm 1. It has four hyperparameters: \( m > 0 \), \( \omega_l \in [0, 1] \) s.t. \( \sum_{l} \omega_l = 1 \), \( \delta \in [0, 1] \) and \( \eta \in [0, 1] \), resp. the initial number of cutting points, the weights for each feature, a similarity threshold under which we do not want to merge two blocks and an improvement threshold under which we consider it is not interesting enough to merge.

Algorithm 1 Split-and-Merge

1: \( \pi_0 \leftarrow \{ \frac{[0, T]}{m} \}_{i=0}^m \)
2: do
3: \( \quad \text{Calculate the family } \{|d^i|_{j=1}^i\} \text{ for } \pi_k \)
4: \( \quad F^i_j \leftarrow \frac{1}{n} \sum_{l=1}^{\pi} \delta_l d^i_j \)
5: \( \quad \text{for } F^i_j \in \{F^i_j | F^i_j \geq F^{i,j}, \forall j_j > j \} \text{ do} \)
6: \( \quad \quad \text{if } F^i_j \geq \delta \text{ and } R(\pi_k, j) \leq \eta \text{ then} \)
7: \( \quad \quad \quad \pi_{k+1} = \pi_k \setminus \{p_j\} \)
8: \( \quad \quad \text{break} \)
9: \( \quad \text{end if} \)
10: \( \text{end for} \)
11: \( \text{while } \pi_k \neq \pi_{k+1} \)

5 RESTRICTING THE SEARCH SPACE

If the audio of the video is intended to describe the video content (e.g. a documentary or a commented video for educational purposes), we make a stronger assumption: a cutting point between two semantic blocks can only occur between two sentences. Not only this assumption turns the continuous problem into a (almost) discrete problem, but it also provides an upper bound on the number of cutting points. If we denote by \( S = \{s_1, ..., s_K\} \) the set of sentences in the perfect transcript, the number of cutting points is bounded by \( K \). It is only almost discrete, because in many cases the time interval between two sentences may be quite long and there is still a need to determine where to cut exactly.

Let \( [s^*(s_j), s^*(s_{j+1})] \subset [0, T] \) denotes the time interval between two consecutive sentences \( s_j \) and \( s_{j+1} \). In other words, \( \forall p \in \pi^*, \exists j \in \{1, ..., K - 1\} \text{ s.t. } p_j \in [s^*(s_j), s^*(s_{j+1})] \).

To take advantage of this assumption, we change the initial uniform partition \( \pi_0 \) defined in Algorithm 1 by \( \pi_0 = \{p_j | \forall j \in \{1, ..., K - 1\}, p_j = s^*(s_j) + \frac{1}{T} \} \). Contrary to the previous version, there is no parameter \( \delta \) to control the (best case) precision. However, as we know there is a unique cutting point in \( [s^*(s_j), s^*(s_{j+1})] \), we can split it into an arbitrary number of potential cutting points and select the one that minimizes the aggregated similarity. This step is done after applying Split-and-Merge as a refinement step.

![Figure 1: Illustration of Split-and-Merge. On top, the initial uniform partition, finer than the optimal partition \( \pi^* \). At the bottom, the partition after few steps: several cutting points have been removed. It is to be noticed that, as \( p_1^* \) or \( p_2^* \) are not on the initial partition, it is impossible to have them in the output of Split-and-Merge.](image)
6 EXPERIMENTS AND RESULTS

The plan of this section is as follows: we present the data we used in Section 6.1, then we elaborate the protocol in Section 6.2. We analyse the results in Section 6.3. Last but not least, in Section 6.4, we address some problems or possible critics of our protocol.

6.1 Videos

We used three short videos (2:16 to 4:00 minutes) available on Youtube:

1. Running an experiment in the IBM Quantum Experience
2. IBM Bluemix - 3 minutes demo (NodeJS, SSO)
3. IBM Watson for Oncology Demo

Apart from being short, the videos have some specific characteristics. The first video presents how to create, run, and access to the result of a quantum algorithm. The video is relatively naturally broken down into parts except the moment when it explains the algorithm where the video switches between displaying a screen with the software to perform the experiment and an illustration using cards. Also, it seems to us that there is a long block in the middle of the video which would require Split-and-Merge to really understand how to remove inappropriate cutting points. The second one is very naturally broken down into several parts by a short notice with the subject of the next part written during the transitions. Between transitions, the speaker is visually present on the video which can perturbate the visual metrics since he tends to move, thus modifying perceptual hash and pixel-based distance more than expected. Finally, the third video has been selected because it is harder to break down since the speaker flow and presentation seem more “continuous” without neat transition between screens or topics. Also, it integrated a visual transition (from black screen to a web page and the contrary) at the beginning and at the end, which can trick both humans and the visual metrics.

6.2 Protocol

We asked people through a public form to indicate their optimal partition with the following instruction: For each video, describe how you would break it into “steps” (a bit like a chapter in a book). We added that in case of doubt, one can indicate that a transition is weak. We respectively collected seven, five and five answers for each video. For each answer, we obtained, we plot the cutting points $t_i$ as a three second interval centered on $t_i$ (in blue for the weak cutting points, red otherwise) such that we can visually observe the areas on the timeline such that people agree there is a cutting point.

Example of answer for the first video:

0:12 W
0:20 W
0:31 W
0:38 W
0:59 W
2:35 W
2:56 W
3:49 W

In a second step, we used Split-and-Merge (without the assumption made in Section 5) on the three videos with the following settings: a uniform initial partition with a length of three seconds per block, $\delta = \frac{1}{n}$, $\theta_i$ (i.e. each feature has the same importance), $\delta = 0.9$ and $\eta = 0.1$. Finally, we reported the partition found under the form of three second intervals centered on the cutting points. The choice to assimilate a cutting point to a three second interval result both from the protocol (people may agree on the same cutting point, but report different time to one or two seconds difference) and from the algorithm (for computational reasons, extracting and processing the screenshots is relatively costly). In the analysis, we consider that two cutting points coincide if there is an intersection between their intervals. In other words, two cutting points are assimilated if they are separated by less than three seconds, which correspond to the selected precision for the algorithm.

6.3 Results

The results are summarized by Figure 2. Let us analyze first the panel answer. For the first video, the cutting points are relatively numerous with 11 in total. A lot of them are mostly in blue indicating a doubt from the panel. Around 1:00, a ten second interval is formed, indicating a variability in the choice of cutting points. This was expected as mentioned previously in the video description. The large block from around 1:00 to 2:25 is also confirmed by the answers. For the second video, and as we expected, there is a neat consensus: every person from the panel answered the same that is to say, putting a cutting point when the transition is visually indicated. There is one exception with a weak transition, in one answer, located on the transition between the black screen at the beginning and the real content of the video. Regarding the third video, the participants seem to agree on whether or not a transition is weak or strong. For the second distinct area in the timeline, around 24s, the union of intervals made of the cutting points has a nine second length. Checking the answer individually confirms that there is no one adding more than one cutting point in this area. The results globally reflect our own personal answers on the cutting points and more important, the panel answers reflect the difficulties we described in the previous section. Globally, the huge amount of weak transitions in the first and last video confirms that the semantic partitioning problem is not well defined as the perception of what exactly is a semantic block varies between persons and videos.

In the first video, only two out of eleven cutting points have been found. There are two pairs of red and blue cutting points separated by only four seconds (in resp. 0:08 and 0:30) and for both, Split-and-Merge identified a cutting point in the middle. We explain this by the fact that the algorithm did not find relevant the blue cutting points (the selected cutting points are closer to the red ones). One good point to notice is the absence of cutting point in the large middle block. In general, the results on this video are disappointing because the video seemed easy to break for humans, and none of the most consensus cutting points (red at 1:10 and 2:28) has been identified. In the second video, the size of the final partition matches the size of the panel optimal partition with seven cutting points. However, except for two points, there are all misplaced: the one around 0:25 is about five second late and the ones 2:20 and 2:40 are three seconds in advance. The two others are clearly misplaced and almost in the worst possible location, i.e. in the middle of an interval (at least not the same). It is quite surprising that the video with the
most clear consensus on the optimal partition does not provide better results. The fact that the speaker appears on the video may be an explanation as between two frames the number of pixels that changed is most likely to be higher without changing the semantics. This would imply that the visual metrics play a more important role than we first expected, at least for videos with this characteristic. Further experiments with different weights $\omega_i$ and higher thresholds for the visual metric need to be carried. In the last video, Split-and-Merge successfully identified seven out of nine cutting points among which, two are identified with a zero second difference. However, the most important cutting point (not identified as weak, and being in the five answers with at most one second difference) is not identified. In the large interval around 0:24, Split-and-Merge identified the cutting point at the extreme right side. Surprisingly, Split-and-Merge performed the best on the third video which seems to the authors to be the hardest to naturally break into semantically distinct parts.

6.4 Discussion on the protocol

The small number of answers collected to create the optimal partition may be a threat of validity. However, as shown in Figure 2, a consensus seems to emerge, especially for the second video which can mitigate this critic. Another drawback in our protocol is that the results are more qualitative than quantitative: we do not measure any distance from the Split-and-Merge result to the optimal partition. We justify this approach by the fact that such metric may be useful to compare algorithms to each others, but too ad-hoc to be interesting by itself. In a preliminary evaluation and taking into account the ’fuzziness’ of the optimal partition, a qualitative description of the results appeared to be enough to us. A last critic that may appear is the lack of clear definition of what is or should be a cutting point in the instruction for the panel. This has been done on purpose since a more precise indication would probably contain explicit references to some feature streams or metrics on feature streams and indirectly influence the answer. A last critic that may appear is the lack of clear definition of what is or should be a cutting point in the instruction for the panel. This has been done on purpose since a more precise indication would probably contain explicit references to some feature streams or metrics on feature streams and indirectly influence the answer.

7 CONCLUSION AND FUTURE WORK

In this paper we presented the Video Semantic Partitioning problem defined as finding an optimal partition w.r.t. an unknown and not so well defined function measuring the semantic. We reformulated the problem as the optimization of a convex combination supposedly approximating the unknown function. We justify this construction by the extraction of feature streams, each of them holding a part of the semantic contained in the video. To solve the problem, we introduced a heuristic called Split-and-Merge. Its main idea is to cut the video in too many pieces before greedily merging the blocks until the improvement is neglectable. Under a restrictive assumption, we reduced the continuous problem to a (almost) discrete problem and adapted the algorithm consequently. Finally, we presented the results obtained on a few test videos and discuss the protocol.

Despite the variability in the quality of results, the results are encouraging taking into account no hyperparameter tuning was performed. Surprisingly, the algorithm obtains the best results on what we considered to be the hardest video and the worst result on the easiest video.

In future work, we will investigate how to tune the hyperparameters to obtain the best out of the algorithm on a set of videos. We are also interested in comparing the convex combination approach presented in this paper with a multi-objective approach. Also, we need to investigate how good the modified algorithm under the restrictive assumption perform compared to the initial version. Last but not least, we plan to incorporate more advanced feature streamed such as objects obtained by video segmentation.

ACKNOWLEDGMENT

The authors warmly thank IBM BTO and IBM Krakow Software Lab for their financial support.

AUTHOR CONTRIBUTIONS

Marcin Janiszewski pointed out the problem and guided its resolution using IBM Watson Services. Krzysztof Jamrog implemented the algorithm, taught IBM Watson Speech-to-Text to increase the transcript quality. Alexandre Quemy conceived the algorithm, wrote the article, selected the data, conceived and run the experiments, analyze the data.
REFERENCES


Study of the applicability of an itemset-based portfolio planner in a multi-market context

Luca Cagliero
Dipartimento di Automatica e Informatica
Politecnico di Torino
Turin, Italy
luca.cagliero@polito.it

Paolo Garza
Dipartimento di Automatica e Informatica
Politecnico di Torino
Turin, Italy
paolo.garza@polito.it

ABSTRACT
Planning stock portfolios for long-term investments is a well-known financial problem. Many data mining and machine learning strategies have been proposed to automatically predict the set of uncorrelated stocks maximizing long-term portfolio returns. Among others, the use of scalable itemset-based strategies has recently been studied. Potentially, they can analyze large sets of historical prices corresponding to thousands of stocks in the worldwide market indexes. However, the current studies are still limited to single markets.

This paper investigates the applicability of itemset-based strategies for planning stock portfolios in a multi-market context. Scaling the analyses towards multi-market scenarios poses a number of research questions, among which the choice of the diversification strategy, the influence of inter-market correlations among stock prices, and the profitability of multi-market strategies compared to single-market ones. This paper aims at answering to the aforesaid questions by considering a state-of-the-art itemset-based approach.

The experimental results show that itemset-based strategies focus the generated portfolios on the outperforming markets. Furthermore, the performance of multi-market strategies with sector-based diversification is on average superior or comparable to single-market ones.

1 INTRODUCTION
Forecasting the stock markets is a well-known financial problem. It entails predicting the future prices of a set of stocks to drive investments in the short-, medium-, or long-term. Predictions are commonly driven by fundamental or technical analyses [8]. The former studies analyze the overall state of a company or a business (e.g., earnings, production, manufacturing), whereas the latter analyze the historical stock prices, which are assumed to reflect all the external influences. Technical analyses often consider both statistics-based indicators, computed on the sampled stock prices, and graphical patterns, recognized from the price time series, that are likely to be related to specific trends [20].

In this work, we focus on the analysis of the historical stock prices to make long-term predictions. The aim is to generate a portfolio consisting of a subset of market stocks whose prices are likely to increase. To spread bets across multiple assets, thus minimizing the losses in case forecasts turn out to be wrong, portfolios are asked to be diversified, i.e., they should comprise stocks from different sectors, markets, or geographical areas [4, 14].

In recent years, the diffusion of machine learning and data mining techniques has prompted the financial sector and the research community to investigate their application to solve the portfolio generation problem. For example, classification and regression algorithms such as Neural Networks [15, 24], Decision trees [2, 18], and Support Vector Machines [5, 12] have been exploited to predict the future stock directions and prices, respectively, based on the values of multiple dependent variables. Alternative strategies entail the use of
(i) Time series analyses, to pinpoint significant temporal trends in continuous stock signals [9, 11, 13, 25],
(ii) Clustering algorithms, to group stocks characterized by similar behaviors [16, 21],
(iii) Pattern recognition techniques, to recognize graphical patterns coming from technical analyses [17], and
(iv) Particle swarm optimization and evolutionary algorithms, to identify the stocks that maximize a given objective function [1, 7].

Itemset mining is an exploratory data mining technique that focuses on discovering recurrent co-occurrences among items in large transactional dataset [3]. For example, let us consider a transactional dataset collecting the baskets of the customers of a market, where each transaction (basket) consists of a set of distinct items. Frequent itemset mining algorithms have been exploited to discover combinations of items that are frequently purchased together. Since items may have different importance within the analyzed datasets (e.g., different prices and purchased amounts) their occurrences in each transaction can be weighted [19].

Recently, in [6] a first attempt to apply itemset mining techniques to generate diversified stock portfolios has been made. Stocks are represented as distinct items in the dataset. A transactional dataset collects the historical stock prices within a time range. Each transaction corresponds to a distinct timestamp within the given time range and contains all the quoted stocks weighted by their price at the corresponding timestamp. According to the data model described above, itemsets represent candidate stock portfolios consisting of sets of stocks of arbitrary size. In [6] the most interesting itemsets are generated and ranked according to the average return of the contained stocks as well to their level of diversification in the portfolio.

The main advantages of itemset-based approaches are (i) the interpretability of the generated model and (ii) the scalability of the extraction algorithms, which can be applied to very large datasets [22]. On the one hand, the interpretability of the mined itemsets allows domain experts to manually explore the top ranked itemsets to make appropriate decisions. On the other hand, the scalability of the itemset mining process makes the portfolio generation process portable to multi-market domains. The algorithm can analyze large stock datasets acquired from multiple markets and automatically recommend diversified worldwide investments with limited human effort. However, to the best of our knowledge, the application of itemset-based strategies in multi-market contexts has not been investigated yet.

© 2018 Copyright held by the owner/author(s). Published in the Workshop Proceedings of the EDBT/ICDT 2018 Joint Conference (March 26, 2018, Vienna, Austria) on CEUR-WS.org (ISSN 1613-0073). Distribution of this paper is permitted under the terms of the Creative Commons license CC-by-nc-nd 4.0.
2 CONTRIBUTION

This paper investigates the applicability of itemset-based strategies for planning diversified stock portfolios in a multi-market context. Extending the scope of the stock data analysis from single markets to multiple ones poses the following research questions:

Choice of diversification strategy. Stocks can be categorized based on different strategies, such as the industrial sector of the underlying company, the market index of the stock, the nationality of the company, or the country/continent associated with the market index. These categorizations can be exploited to diversify investments across uncorrelated assets. In multi-market contexts, the choice of the diversification strategy is not trivial, as it could relevantly affect the performance of the stock portfolio planner. The research questions we would like to address in this study can be formulated as follows: Which type of diversification strategy better preserves the portfolio profits? Which type of diversification strategy allows optimally spreading investments across multiple assets?

Influence of inter-market stock correlations. Studying the correlation between the prices of multiple stocks is crucial for professional traders and private investors to take appropriate decisions. However, analyzing the influence between the stocks belonging to multiple markets is potentially challenging, because the number of stocks indexed in worldwide markets is very large. Itemset-based approaches [6] allow domain experts to set the desired levels of average return and diversification according to the chosen stock categorization. Under these constraints, a set of candidate portfolios is generated. The top ranked portfolios include the stocks with maximal average return and with a diversification level at least equal to the set least diversification level. Therefore, the outperforming markets are likely to be overweighted, while the stocks indexed in under-performing markets are likely to be under-weighted. The research question we would like to address in this study can be formulated as follows: Does the majority of the portfolio stocks belong to outperforming markets? Are the stocks in the portfolio correlated in terms of membership index?

Comparison between different scenarios. Extending the scope of the analysis towards multiple markets gives professional traders and private investors new opportunities of investment on foreign markets. Considering a larger number of considered stocks not only simplifies the process of diversification of the investments, but also allows traders to move investments towards most profitable markets. However, a quantitative evaluation of the benefits for itemset-based approaches of considering multiple markets at the same time compared to single-market analyses is still missing. The research question we would like to address in this study can be formulated as follows: Are the portfolios generated from multiple markets more profitable than those generated from single markets?

In this study we investigated the use of different diversification strategies in multi-market scenarios to gain insights into the effectiveness of itemset-based strategies on large stock data. Furthermore, we analyzed the generated portfolios to understand to what extent inter-market stock correlations are considered in the recommended portfolios. Finally, we empirically compared the performance of single- and multi-market recommendations in different scenarios.

The rest of the paper is organized as follows. Section 3 summarizes the main steps of the diversified stock portfolio planner [6]. Section 4 describes the experimental design. Sections 5, 6, 7 discuss the choice of the diversification strategy, the influence of inter-market stock correlations, and the comparison between multiple and single market strategies, respectively. Finally, Section 8 draws conclusions and discusses the future research perspectives of this work.

3 THE DIVERSIFIED STOCK PORTFOLIO PLANNER

DISPLAN (Diversified stock portfolio planner) [6] is an itemset-based strategy for generating diversified stock portfolios based on the analysis of historical stock prices. It relies on the following steps:

Stock data collection and preprocessing. This step focuses on crawling historical stock prices and collecting them into a unique dataset. It takes as input a list of stocks and a time range. It acquires the daily closing prices of all the considered stocks and stores them into a weighted transactional dataset. Each row in the dataset (called transaction) corresponds to a different timestamp and contains the prices of all the considered stocks at the corresponding timestamp. Each pair (stock, price) occurring in the dataset is denoted as weighted item.

Weighted itemset mining. This step analyzes the correlations between stock prices based on weighted itemset mining techniques [19]. It takes as input the weighted transactional dataset prepared at the previous step and a taxonomy aggregating stocks into higher-level categories. For example, to each stock the corresponding industrial sector can be assigned.

It extracts interesting patterns, called frequent weighted itemsets, from the weighted transactional dataset. A weighted itemset is a set of stocks of arbitrary length, which represents a candidate stock portfolio. The extracted weighted itemsets satisfy the following properties:

(i) The average daily return of all the stocks in the itemset is above a given minimum return threshold minret.
(ii) The percentage of stocks belonging to different categories (according to the input taxonomy) is above a given diversification threshold mindiv.

Portfolio generation. This step analyzes the extracted itemsets to identify the best candidate stock portfolios satisfying all the user requirements. To make the extracted patterns promptly usable by investors for stock portfolio planning the mined itemsets are first ranked in order of (i) decreasing length (i.e., number of contained stocks) and (ii) average daily return. The top ranked itemsets are deemed as the most appropriate hints for buy-and-hold (long-term) investors. More specifically, the itemsets containing the maximal number of stocks are selected as best candidate portfolios, because they satisfy all user requirements (by construction) and contain the maximal number of stocks thus allowing investors to spread their bets over the largest number of different assets. In case of ties, the itemset with maximal least average return is considered as the best candidate stock portfolio because it achieved maximal profit on historical data. On equal terms (i.e., same length and average daily return) the analyst is asked to decide which itemset is deemed to be the most appropriate stock portfolio to consider based on his personal judgment and experience.
4 EXPERIMENTAL DESIGN

We analyzed stock data acquired by means of the Yahoo! Finance APIs [23]. To crawl data, we performed several API requests to retrieve the closing stock prices of several stocks from different market indexes. Each request produces a different stock dataset, which consists of the closing prices of the requested stocks within the considered time period sampled at the desired frequency. In our analyses, we considered the daily closing prices of the stocks in two representative years, i.e., 2008 and 2013. Year 2008 is representative of an unfavorable condition for worldwide financial markets, i.e., the rise of the global financial crisis, whereas year 2013 represents a favorable market condition, i.e., the boom of U.S. markets. Analyzing opposite market scenarios allows us to perform a fair assessment of the portfolio generator with different settings.

To study the performance of the itemset-based portfolio planner in a multi-market context, we classified market indexes based on the corresponding opening timezone as follows: (i) Europe, with approximate opening times from 9am to 5.30pm CET; (ii) Asia and Oceania, with approximate opening times from 2am to 9.30am CET; (iii) North and South America: with approximate opening times from 3pm to 10pm CET. As European indexes we considered the following ones: the Brussels Stock exchange (BEL 20) (20 stocks), the Paris market (CAC 40) (40 stocks), the London Financial Times Stock Exchange (FTSE 100) (98 stocks), the Italian stock exchange (FTSE MIB 40) (40 stocks), the General Athens Composite Index (GD) (59 stocks), the HDAX Deutscher Aktien index (GDAXI) (109 stocks), the OMX Stockholm 30 (OMX) (25 stocks), and Oslo Bors All Share Index (OSEAX) (127 stocks). As Asian and Oceania Indexes we considered the following ones: the BSE Sensex (BSESN) Based on the Bombay Stock Exchange (India), the FTSE Straits Times Index (STI) based on the Singapore Exchange (29 stocks), the Hang Seng Index (HSI) based on the Hong Kong Stock Exchange (50 stocks), the NIFTY 50 (NSEI) consisting of companies listed on the Bombay Stock Exchange (BSE-India) (50 stocks), the NZX 50 (NZS50) based on the New Zealand Stock Exchange (NZSX) (39 stocks), the S&P/ASX 200 (AXJO) based on the Australian Securities Exchange from Standard & Poor’s (199 stocks), and the Taiwan Capitalization Weighted Stock Index TAIEX Index (TWII) (898 stocks). As North and South America Indexes we considered the following ones: the Brazil Broad-Based Index (IBRA) (116 stocks), the US Dow Jones Industrial Average (DJI) based on the New York Stock Exchange (30 stocks), the Brasilian Indice Bovespa (BVSP) (63 stocks), the IPC Index (MXX) based on the Mexican Stock Exchange (34 stocks), the IVBIX 2 Brasilian Index (IVBIX) (50 stocks), the MERVAL Index (MERV) based on the Stock Exchange of Buenos Aires (Argentina) (12 stocks), the US Nasdaq Stock Market index NASDAQ-100 (NDX), the US S&P 500 (GSPC) (502 stocks), and the Canadian S&P/TSX Venture Composite Index (SPCDNX) (338 stocks). The Europe timezone comprises 518 stocks, the Asia and Oceania timezone 1118 stocks, while the North and South America 1147. Hereafter, we will denote as market-based categorization the stock categorization based on the considered indexes. We considered also a sector-based stock categorization according to the Industry Classification Benchmark (http://www.icbenchmark.com/).

To simulate long-term stock investments, we applied the following procedure: (i) We trained the itemset-based model and generated the diversified stock portfolio on the first 7-month time period. (ii) We tested the model by virtually buying the stocks at the beginning of August and selling the whole portfolios in the following year. We varied the selling date in the date range between August, 1st 2013 and August, 1st 2014.

We simulated both long and short selling investing positions. Long selling entails buying the stocks because its price is likely to increase thus yielding a profit in case the price has increased when the stock is sold. Conversely, short selling is the practice of selling stocks that are not currently owned, and subsequently repurchasing them at the end of the investment. If the price decreases, the short seller profits, since the cost of (re)purchase is less than the proceeds received upon the initial (short) sale. Conversely, the short selling position closes out at a loss if the stock price rises prior to repurchase [10].

In the following section we will analyze also the impact of the algorithm parameters on the quality of the training models. Notice that the portfolio profits are lower bound estimates of the actual profits as stocks may be sold one by one rather than altogether and the investments can be reconsidered during the whole period (not only at the end). Furthermore, payoffs produced by intermediate sells can be reinvested during the same time period. Finally, transactions costs and local taxes and fees were not considered.

5 CHOICE OF THE DIVERSIFICATION STRATEGY

We compared the performance of the itemset-based stock portfolio planner (hereafter denoted as DISPLAN for the sake of brevity) by using two alternative diversification strategies:

(i) a market-based strategy, where stocks are picked from different indexes to limit intra-market stock correlations (independently of the industrial sector of the considered stocks), and

(ii) a sector-based strategy, where stocks picked from different sectors (independently of the underlying market index).

As representative examples, in Figures 1, 2, and 3 we plotted the relative returns achieved in year 2013 by the portfolio generated by DISPLAN on the markets of the Asia and Oceania, Europe, and North and South America timezones with both market- and sector-based diversification. For all the considered timezones we reported the results achieved by using long selling positions (see Section 4) and by setting the minimum diversification threshold to 70% (i.e., at least seven stocks out of 10 must belong to different categories). The minimum return threshold values enforced to discard non-profitable sets of stocks are 11%, 9%, and 12%, respectively. To compare DISPLAN performance with that of the considered indexes for each configuration, we plotted also the percentage variation of the benchmark indexes as well as those of an aggregate index consisting of all the underlying indexes the same timezone.

The achieved results show that applying a sector-based diversification yields significantly higher profits compared to applying a market-based strategy (e.g., in the Asia and Oceania timezone on July, 1 2014 the average variation of the sector-based strategy is 12% vs. the 18% of the market-based one). The motivations behind the achieved results are the inherent sparsity of the market-based categorization compared to the sector-based one and the stronger influence of sector-driven stock price movements. Specifically, by applying sector-based diversification within each category the algorithm can choose among a quite large number of stocks. Among the per-sector candidate stocks, some of them are likely to outperform the benchmarks. Therefore, the stocks that under-perform the corresponding sector can be discarded. Conversely, to satisfy the least diversification level, the
market-based strategy could be forced to pick stocks that underperform the corresponding markets index as well. Furthermore, the number of candidate indexes per timezone is still limited (8 for Europe and Asia and Oceania, 10 for South and North America). Thus, in large portfolios to achieve high diversification levels the stocks belonging to low-performing indexes cannot be neglected. On the other hand, the intra-sector correlations among stock prices appear to be stronger than intra-market ones. For example, a drop of the oil price negatively influences all the correlated stocks independently of the market index. In summary, based on the achieved results it turns out that considering a stock categorization based on sectors prevents the DISPLAN algorithm from making inappropriate decisions.

6 ANALYSIS OF INTER-MARKET STOCK CORRELATIONS

The stock portfolios generated by the DISPLAN algorithm may include stocks belonging to multiple markets. Hence, it is interesting to investigate how the inter-market correlations among stock prices could affect the performance of the DISPLAN algorithm.

Figure 4 shows, as representative case study, the relative returns achieved in year 2013 by the portfolio generated by DISPLAN (with long selling position) in both multi- and single-market scenarios. Specifically, as representative study, we reported the performance of the portfolio generated by DISPLAN from the analysis of the stock data related to all the markets in the Asia and Oceania timezone as well as the performance of the portfolios generated by DISPLAN from each index in the same timezone. Notice that since the AXJO, HSI, and STI indexes did not produce any single-market portfolios satisfying the minimum return and diversification constraints, the corresponding curves were omitted.

By enforcing a minimum diversification level among sectors of 70% ($\text{mindiv}_v=70\%$), the multi-market portfolio consists of the same stocks selected by the best performing single-market portfolio, i.e., the one generated for the TWII index of Taiwan (see Figure 4(a)). Conversely, while setting the highest possible value of sector-based diversification level ($\text{mindiv}_v=100\%$), all the stocks in the portfolios must belong to a different sector. The maximally diversified portfolio differs from the former one because a stock from the AX index (JBH) was selected. The sector of stock JBH under-performed the benchmark sector (approximately +15%). For this reason, despite the higher diversification of the new portfolio its relative returns are still relatively high. On the contrary, by setting the diversification threshold to its maximal value the portfolio generated from the single-market TWII index significantly decreases its relative returns, because the added stocks under-performed the benchmark sector index (see Figure 4(b)).

7 COMPARISON BETWEEN MULTI- AND SINGLE-MARKET STRATEGIES

We compared multi- and single-market strategies to assess the applicability of the DISPLAN system in a multi-market scenario. The results, which were summarized in Figure 4 for a representative case study (Asia and Oceania, long selling position, year 2013), show that the the multi-market approach performed better than most single-market strategies while it performs as good as the best performing single-market one. Therefore, applying the itemset-based portfolio generation strategy is particularly appealing, as it allows us to diversify investments across multiple market indexes without significantly degrading the portfolio returns.

Figure 5 shows the performance of the DISPLAN algorithms by setting a relatively high minimum return threshold (16%). The results show that the multi-market strategy performed as good as the best single-market strategy (the one corresponding with the best performing index). Conversely, many single-market strategies appear to be less effective because few candidate stocks are selected. The motivation behind is that, given a large number of candidate stocks from multiple markets, the likelihood that a set of highly profitable stocks diversified over sectors is found is higher. The more stocks the algorithm can analyze, the most likely a profitable and diversified stock portfolio can be discovered. To avoid data overfitting the number of stock data samples should be at least on the order of the number of analyzed stocks.

8 CONCLUSIONS AND FUTURE WORK

In this paper, the application of itemset-based approaches to generating diversified stock portfolios in a multi-market scenario has been studied. Given a stock categorization and a dataset collecting the historical prices of a potentially large set of stocks, itemsets representing profitable yet diversified stock portfolios can be automatically extracted and recommended to investors, professional and not. The scalability of itemset-based techniques prompted their application in a multi-market scenario, where the following issues have been addressed.

(i) The choice of the diversification strategy is not immediate, because in multi-market contexts investors could spread bets across either markets or sectors. Based on the achieved results, sector-based diversification yielded significantly better results due to the good balancing between the stocks across sectors.

(ii) The inter-market correlations among stocks are properly handled by the itemset-based strategy, as the most profitable stocks are selected independently of the underlying market index.

(iii) Multi-market strategies performed better than or as good as single-market ones in most of the performed experiments.

Future works will entail applying itemset-based strategies on datasets collecting historical stock prices at finer time granularities. The aim is to apply itemset-based models to drive medium- and short-term investments (e.g., intra-day trading). Furthermore, we will try to apply more advanced itemset mining techniques, such as utility and probabilistic itemset mining, in order to (i) shape investments according to the amounts of stocks already in the portfolio, and (ii) take stock volatility and risk levels into account.

REFERENCES


Figure 1: Market- vs sector-based diversification: Average percentage variation w.r.t. August, 1 2013. Asia and Ocean markets, long strategy, \( \text{mindiv}=70\% \), \( \text{minret}=11\% \)

Figure 2: Market- vs sector-based diversification: Average percentage variation w.r.t. August, 1 2013. Europe timezone. Long selling position. \( \text{mindiv}=70\% \), \( \text{minret}=9\% \)

Figure 3: Market- vs sector-based diversification: Average percentage variation w.r.t. August, 1 2013. North and South America timezone. Long selling position. \( \text{mindiv}=70\% \), \( \text{minret}=12\% \)
Figure 4: Multi-market vs single-market: Average percentage variation w.r.t. August, 1 2013. Asia and Oceania timezone. Long selling position. minret=11%.

Figure 5: Effect of high minimum relative return threshold: average percentage variation w.r.t. August, 1 2013. Asia and Oceania timezone. Long selling position. minoc=70%, minret=16%.


Mining implicit data association from Tripadvisor hotel reviews

Vittoria Cozza
Department of Information Engineering, University of Padua, Italy
vittoria.cozza@dei.unipd.it

Marinella Petrocchi
IIT-CNR, Pisa, Italy
marinella.petrocchi@iit.cnr.it

Angelo Spognardi
Dipartimento di Informatica, Sapienza Università di Roma, Rome, Italy
spognardi@di.uniroma1.it

ABSTRACT
In this paper, we analyse a dataset of hotel reviews. In details, we enrich the review dataset, by extracting additional features, consisting of information on the reviewers’ profiles and the reviewed hotels. We argue that the enriched data can gain insights on the factors that most influence consumers when composing reviews (e.g., if the appreciation for a certain kind of hotel is tied to specific users’ profiles). Thus, we apply statistical analyses to reveal if there are specific characteristics of reviewers (almost) always related to specific characteristics of hotels. Our experiments are carried out on a very large dataset, consisting of around 190k hotel reviews, collected from the Tripadvisor website.

1 INTRODUCTION
Social media, forums, and blogs are privileged vehicles for posting and spreading online reviews. Among the goods and services that are discussed every day on the Internet, we find those belonging to the most disparate categories, like, e.g., food, clothes, music, toys, etc. Particularly, the practice of choosing and booking preferred destinations has been greatly eased by the possibility for users to consult previous feedback about hotels and restaurants. According to comScore Media Metrix\(^1\), Tripadvisor is the world’s largest travel e-advice site, providing advices as reported by actual travellers. Tripadvisor counts more than 87 million visitors per month\(^2\).

Not only common users, but also service providers have strong motivations to analyse the myriads of posts, tweets, and comments available online. The latter will benefit by adjusting, e.g., their products lines and advertisement campaigns, while the former by relying on previous experiences for addressing their needs and matching their expectations. Furthermore, online reviews are a precious source of information, e.g., to unveil implicit and/or unexpected characteristics of the reviewers. As an example, in [13] the authors investigate if and how the words —and their use— in a review are linked to the reviewer’s gender, country, and age.

In [8], the authors present a novel approach to build feature-based user profiles and item descriptions by mining user-generated reviews. Such additional information can be integrated into recommender systems to deliver better recommendations and an improved user experience.

In our previous work [9], we exploited a Tripadvisor dataset in order to investigate how subjectivity of reviewers affects the scores assigned to hotels. Thus, we leverage sentiment analysis techniques to identify mismatches between the text and the score in online review platforms.

Since several aspects can influence the customer experience (e.g., the hotel price, or the presence of restaurants, cafe, discos in the hotel neighborhood, the connections with bus/train stations and airports, etc.), in this work we propose an automatic approach - based on association rules - to understand which factors most influence consumers’ reviews. We consider a very large dataset consisting of around 190k hotel reviews collected from Tripadvisor, enriching the dataset by extracting a series of hotel-centric and reviewer-centric features. We leverage these features to list correlations among hotel properties, reviewer’s characteristics, and the review score. The results are obtained applying association rules techniques to our dataset. Findings are both expected - such as that the hotels close to entertainment and food areas are ranked with the highest scores - and less intuitive - such as that those reviewers featuring a very low activity (measured with a lower bound in term of given reviews), considering their stay in a particular area, select - very often - hotels with a low number of transportation means in the neighbourhood.

We argue that, with our approach, sociologists and marketing experts could analyse the results of the association rules to better understand some extra reviewers’ characteristics and connections with the reviewed service. This kind of analysis paves the way for surveying a larger segment of the population than that usually interviewed through standard polls.

2 DATASET
To conduct our study, we grounded it in a dataset composed of real reviews taken from the Tripadvisor\(^3\) website. In particular, our dataset contains all the reviews that can be accessed on the website between the 26th of June 2013 and the 25th of June 2014 – date of the newest extracted review – for hotels in New York, Rome, Paris, Rio de Janeiro, and Tokyo. With a straightforward approach, we were able to collect the following pieces of information for each review:

- the review date, text, and numeric score;
- the reviewer username, location, and triptype, being the type of trip, one among the following five categories: Family, Friends, Couple, Solo Traveler, and Businessman;
- the ID of the hotel which the review refers to.

In addition to the above elements, we collected from Tripadvisor all the hotels of the considered reviews and included in our review dataset some additional data regarding the reviewed hotels. In particular, leveraging the ID of the hotel which the review refers to, we have gathered:

- the hotel name and full address (where full address includes the street address, the city, and the country);

\(^1\)https://www.comscore.com/Products/Audience-Analytics/Media-Metrix - All sites last accessed December 23, 2017.
\(^3\)http://www.tripadvisor.com
• the category of the hotel (number of stars);
• the number of guest pictures for the hotel.

It is worth noting like the above lists are not exhaustive, i.e., they do not represent all the information accessible from Tripadvisor. As an example, further information available for a review are the scores assigned by reviewers to specific aspects of a hotel, like location, cleanliness, sleep quality, rooms, and service. However, for the scope of the current work, we focus on those summarised for the reader’s convenience in Table 1. We exploited such pieces of information to further expand the dataset, with enriched features, as described in the next Section 2.1.

<table>
<thead>
<tr>
<th>Basic information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Review</td>
</tr>
<tr>
<td>Date</td>
</tr>
<tr>
<td>Text</td>
</tr>
<tr>
<td>Score</td>
</tr>
<tr>
<td>Reviewer username</td>
</tr>
<tr>
<td>Reviewer location</td>
</tr>
<tr>
<td>Triptype</td>
</tr>
</tbody>
</table>

Table 1: Considered information in the basic dataset

We have discarded reviews by “Anonymous” users, since they represent users of the platform www.daodao.com—the Chinese version of Tripadvisor—where all the reviewers are indifferently grouped in this single virtual username. We have further limited our analysis on reviews whose textual part is in English, following the language identification and analysis approach presented in [5]. While the reviews accessible from Tripadvisor in the year under investigation are 353,167, after the pre-processing the resulting dataset is made up of 189,304 reviews in English, provided by 142,583 Tripadvisor’s registered users that reviewed 4,019 hotels. Table 1 recapitulates the information extracted from the dataset, while Table 2 shows the distribution of the reviews per given score value. As shown, the values distribution is highly unbalanced, being the highest score the most frequent in the dataset (reflecting indeed the distribution usually featured by review platforms).

<table>
<thead>
<tr>
<th>Rating Value</th>
<th>Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6,504</td>
</tr>
<tr>
<td>2</td>
<td>8,826</td>
</tr>
<tr>
<td>3</td>
<td>24,627</td>
</tr>
<tr>
<td>4</td>
<td>64,949</td>
</tr>
<tr>
<td>5</td>
<td>84,398</td>
</tr>
</tbody>
</table>

Table 2: Distribution of the given scores in the dataset

2.1 Hotel-centric and reviewer-centric features

Starting from the information collected in the basic dataset, we have augmented it performing some further elaboration. In particular, we enriched the data regarding the reviewed hotel with the following features:

• the popularity, defined as the number of reviews for a given hotel. While we have neither the list of actual bookings available, nor Tripadvisor requires the reviewer to show a proof to have been a guest in the hotel, this feature, when computed on a large number of reviews per hotel, could indirectly act as a quantification of the actual hotel clients;
• the hotel triptype, defined as the most frequent reviewer triptype for a given hotel (whereas triptypes are Families, Friends, Couples, Solo Travelers, and Businessmen);
• the geospatial coordinates (latitude and longitude);
• three points of interest (POI) features, defined as the number of transportation services, restaurants, and attractions, respectively, in a range of 300 meters around the hotel.

Popularity and Hotel triptype have been computed looking at how many and which kind of reviewers have reviewed the hotel. The geospatial coordinates have been calculated with Google Places APIs, starting from the hotel name and full address. Then, latitude and longitude, together with the parameter “radius=300”, have been given as input to the Google Radarsearch API to find the number of points of interest (POI) related to transportation, food, and entertainment.

The data regarding a reviewer, instead, have been enriched with the following features:

• the reviewers’ activity, defined as the number of reviews they have written (under the observation period). Our intuition is that this feature could be useful to discriminate between frequent travelers and sporadic ones.
• the gender of the reviewer. This feature has been extracted with the Namsor Onomastics machine learning tool, able to recognise the language behind a name, thus identifying the gender according to that language vocabulary with high accuracy [4].

After cleaning the username from numbers and symbols and splitting it in two parts (where one is likely to be the name and the other one, when available, the surname), we have called the “onomastics/api/json/gendre” API. This service takes as input name and surname and returns the recognised gender. We have used regular expressions to clean the username from symbols and numbers and for splitting the username. This was possible since, in many cases, the name and surname were separated by a space, or the surname started with an uppercase letter. Some examples of username are: "Eldon S", "MeganJones88".

Unfortunately, for a subset of reviewers, it was not possible to derive the gender from their usernames. This happened for 9,507 reviewers (corresponding to 6% of the entire reviewers set), which wrote 12,653 reviews. Examples of usernames for which it was not possible to derive the gender are Hope-and-Dreams, mistyrabbit, A.TripAdvisor Member, R W, E A, Nickeykol, NawakRed, FreeTravel81. We labeled with unknown the gender of such 9,507 reviewers.

<table>
<thead>
<tr>
<th>Hotel ID</th>
<th>Triptype</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

Hereafter, we will refer to this dataset as the basic dataset. Indeed, in the following, we will extract hotel-centric and reviewer-centric features to enrich the basic set (see Section 2.1).
3 ASSOCIATION ANALYSIS

Association rule mining is a well-known and widely applied methodology for discovering frequent patterns, correlations, and causal structures in transaction and relational databases, as well as in other information repositories [12]. Thus, given a set of items (or itemsets), association rule mining allows to define rules predicting the occurrence of an item (or more), given the occurrence of other items in the same itemsets.

A popular application is basket data analysis, where itemsets are transactions, representing lists of items in the consumers’ baskets. An example of transaction is: {Bread, Steak, Juice, Butter, Chips, Beer}. When several others are collected, e.g., in a large database, the methodology allows to automatically find associations like, e.g., {Bread} ⇒ {Steak} (steaks are often purchased with bread). Beside sales transaction analysis, basket analysis can be applied to other situations like click stream tracking, spare parts ordering and online recommendation engines - just to name a few7.

An association rule (AR) is generally defined as an implication expression of the form $X \Rightarrow Y$, where $X$ and $Y$ are disjoint itemsets. They represent, resp., the condition and the consequence of the rule.

The strength of an AR is commonly measured through the two metrics support and confidence. Support gives the fraction of itemsets in the dataset that contains both $X$ and $Y$. Confidence says how frequently items in $Y$ appear in itemsets that contain $X$. As an example, we want to know the strength of the rule [Bread] ⇒ [Steak] in a dataset with 100 transactions, corresponding to 100 consumers’ baskets. Suppose that itemset [Bread, Steak] occurs 30 times, and that itemset [Bread] occurs 40 times, than the support of the rule is equal to $\frac{30}{100}$, while its confidence is $\frac{20}{30}$.

As discussed in [3], rules with high values for confidence and support do not always correspond to meaningful ARs, especially when working with real datasets, due data can be unbalanced.

For example, one rule could have a very high confidence, but only due to the fact that the item in the consequence is very frequent. In this case, the rule is not relevant. Instead, one rule could have a low confidence, due to the fact that the item in the consequence is very unfrequent in general, but it could still be relevant. Considering the above observation, to evaluate the statisticall significance of the ARs, two other metrics are often used: lift and conviction.

Lift is defined as the confidence divided by the support of the consequence:

$$lift(X \Rightarrow Y) = \frac{\text{supp}(X \cap Y)}{\text{supp}(X) \cdot \text{supp}(Y)}$$

(1)

With respect to confidence, the lift measures the importance of the association considering also the dependence from the support of the consequence.

Conviction is defined by the ratio of the frequency of itemsets that don’t contain the consequence, to the frequency of incorrect predictions:

$$\text{conv}(X \Rightarrow Y) = \frac{1 - \text{supp}(Y)}{1 - \text{conf}(X \Rightarrow Y)}$$

(2)

Both lift and conviction values ranging over the (0,1) interval mean negative dependence, values above 1 mean positive dependence, and a value of 1 means independence.

When items are also divided according to different classes, it is possible to force the AR analysis to return a specific class in the consequence. The obtained rule is called “class association rule” (CAR). The CAR is an implication of the form:

$$X \Rightarrow y, \text{ where } X \subseteq I \text{ and } y \in Y$$

(3)

where $I$ stands for the itemsets and $Y$ for the classes. The definition of the aforementioned metrics holds also for CARs.

The $a$ priori algorithm [2, 16] is one of the most popular algorithms to find frequent itemsets, i.e., itemsets whose support $\geq$ minsup.

In this work, we apply the association rule mining to the hotel reviews scenario. Each itemset corresponds to a distinguished review, and it is a vector whose components are the values of the features extracted and detailed in Section 2. The same features are reported in Tables 4, 5, 6 for the reader’s convenience, together with additional information that are useful here. CARs analysis can be applied when considering also the class, that in our scenario corresponds to the review score, a discrete value with a range between 1 and 5.

To enable the application of the $a$ priori algorithm, we have first discretised those features that natively ranged over a large set of values. As an example, in Table 3, a very low label for Guest Pictures indicates a hotel with a number of pictures comprised from 0 to 11. Still in that table, a medium label for Popularity means a hotel that has been reviewed $n$ times, where $n$ ranges over [433, 1156]. The values in Table 6 should be read as follows: looking at the first line of the “Geo Food” part of the table, our review set contains 37,851 reviews about a hotel, which has a number of restaurants in the range [0, 37] within a radius of 300 mt. Indeed, many different reviews are on the same hotels, being the number of hotels reviewed equal to 4,019, see Section 2.

All the tables also report the Frequency indication, i.e., how many reviews correspond to those values for those features, with respect to the values and features in the tables (still quite obviously, the sum on the values in the Frequency column equals to the total number of reviews considered, 189,304).

http://pbpython.com/market-basket-analysis.html

---

Table 3: Hotel-centric and reviewer-centric features augmenting the basic TripAdvisor dataset

<table>
<thead>
<tr>
<th>Features</th>
<th>Reviewer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hotel Reviewer</td>
<td>Activity</td>
</tr>
<tr>
<td>Popularity</td>
<td>Gender</td>
</tr>
<tr>
<td>Hotel Triptype</td>
<td>Geospatial Coordinates</td>
</tr>
<tr>
<td>Points of Interest</td>
<td></td>
</tr>
</tbody>
</table>

7 The sum on the values in the Frequency column equals to the total number of reviews considered, 189,304.)
In order to find ARs and CARs, we applied the Weka framework [11] implementation of the *a priori*. The Weka *a priori* implementation allows to rank the rules according to different metrics. Among them, we rely on confidence, lift, and conviction. For AR analysis we generate a large number of rules with lift above 1. For CAR analysis, we generate a large number of rules with confidence above 0.2 and then we compute the lift (since, for CAR, Weka does not natively include the ranking based on lift). We finally select the rules with lift greater than 1.

Both for the generated ARs and CARs, we then manually select the most interesting rules, among those with the highest lift and conviction. Table 7 and Table 8 report an excerpt of the results for both scenarios.

### 3.1 Discussion

Association analysis results are reported in Table 7 and Table 8, please notice we only consider those rules that lead to a lift and conviction greater than 1. It is worth noting like \(|X \cap Y|\), when divided by the size of the dataset, corresponds to the support of the given rule.
We summarise the main findings, as follows. Rule $r_1$ states that those reviewers featuring a very low activity, considering their stay in France, select - very often - hotels with a low number of transportation means in the neighbourhood. The rule holds for 19,199 reviews, over a total of 29,837 reviews, with equal premises. Rule $r_2$ states that males visiting US prefer hotels with a high popularity. Rule $r_7$ states that, when the hotel has low transportation means in the neighbourhood, and the number of stars for that hotel is unknown (this may corresponds to accommodation facilities like hostels), its rating is equal to 3. Rule $r_{10}$ states that Japanese people staying in a 3 stars hotels rate those hotels with a score equal to 4. Rule $r_{14}$ in Table 8 states that hotels close to entertainments, which are 37,998, are scored with the top score 5 the 50% of times.

This kind of study provides a general approach for a preliminary data exploration. While the explanation for certain rules is very intuitive, well-grounded justification for others is left to experts in the field. We argue that this kind of analysis corresponds to a preliminary step, useful for suggesting which extra-features could be exploitable to build an enhanced hotel recommendation system. Also, we acknowledge that the analysis is based on the available (direct or indirect) information, obtained from the TripAdvisor’s website. More detailed features could consider elements like price or number of guests. This would allow to obtain other interesting rules, which remain an exclusive prerogative of the hoteliers.

4 RELATED WORK

E-advice technology offers a form of "electronic word-of-mouth", with new potential for gathering valid suggestions that guides the consumer’s choice. Extensive and nationally representative surveys have been carried out in the recent past, "to evaluate the specific aspects of ratings information that affect people attitudes toward e-commerce". It is the case, e.g., of work in [10], which highlights how people, while taking into accounts the average of ratings for a product, still do not take care of the number of reviews leading to that average. Recent work showed that, instead of showing first to the users the reviews with the highest scores, a different order, based, e.g., on the user profile, could be considered [8]: that work integrates new features based on the user profile into recommender systems, to deliver better recommendations and provide an improved user experience. Similarly, in [19], the authors focus on score values given by previous contributors whose preferences are close to the user’s preference. Even almost one decade ago, the work in [1] applies text mining tools to online reviews to define rules sets, to identify contextual information in the texts, which goes beyond a mere order of numerical scores. Similarly to our work, they rely on TripAdvisor, focusing however on text analysis only.

However, the cited literature proposes systems that recommend a service based on the intrinsic characteristics of that service (e.g., characteristics of the hotel and its facilities). Other works, similar to ours, investigate if, and how, the review data hide social and/or economic information of the reviewers. One example is mining reviews to exploit them as a textual resource for sociolinguistic studies at a large-scale, as done in [13]. This work leverages the size of the reviews corpus as a more statistically solid base for the analysis, with respect to manually-collected corpora. Since reviews sites, such as Trustpilot\(^8\), may contain reviewer metadata like, e.g., age, gender and location, the work highlights gender-specific lexical differences, the the distribution of regional markers, spelling variations and the use of grammatical constructions across the reviewers.

The work from [17], which focused on reviews manipulation, exploits reviewer-centric and hotel-centric features to identify outliers: the work compares hotels reviews and related features across different review sites, outperforming the detection of suspicious hotels with respect to check the reviews on sites in isolation. Relying on visualization tools, the authors of [6] highlight suspicious changes on reviews scores, while work in [7] proposes new score aggregators to let review systems robust with respect to injection of fake scores.

Research effort has also being spent to understand which are the factors that let a review perceived as useful: in [15], the authors highlight how the reviewer history is a dominant factor to let a review be voted as useful or not. In [14] propose to use the reviews as a source for demographic recommendations.

In this work we enhance the review dataset with additional features based on characteristics of the reviewer (e.g., gender) and the hotel (e.g., popularity and the neighbourhood). On the contrary, work in [18] studies how, independently from the type of service or the type of reviewer, the scores may be affected by external factors, such as the whether conditions and the daylight length of the service cities. We leverage an extensive experimental campaign, addressing around 190k real reviews, which leads to the provision of statistically sound results. Addressing a large scale of data has been done also in [13], which already has targeted users’ reviews as a rich source of information for sociolinguistic studies. While they achieve correlations between metadata in the reviews’ profile and the review text to let writing styles emerge, we highlight association evidence among hotels and reviewers features and the reviewer’s attitude to score the hotel.

5 CONCLUSIONS

We focused on hotel reviews to investigate which factors could impact the scores that reviewers assign to hotels throughout the world. First of all we have enriched review data with with novel hotel-centric and reviewer-centric features, obtained for example through linked data information available from the web, then we have applied association rule mining to focus on these features possibly motivating the classification scores.

The approach can help both consumers and providers: the former could achieve a better awareness on how to read the reviews (consumers), the latter on how to improve their services (providers). The providers also can query a very large segment of population, in an automatic way and without relying on standard interviews.

The proposed technique is also applicable to a various range of services: accommodat, car rental, food services, to cite a few. Being association rule mining parametric with respect to the itemsets in input, the approach is easily extensible to further features not considered here, such as, e.g., the service price.

6 ACKNOWLEDGMENTS

This research is partly supported by the EU H2020 Program, grant agreement #675320 (NECS: European Network of Excellence in Cybersecurity). Funding has also been received by Fondazione Cassa di Risparmio di Lucca that partially finances the regional project RevieLand. Vittoria Cozza is also supported by the Starting Grants Project DAKKAR (DAKa benchmarkK for Keyword-based Access and Retrieval) promoted by University of Padua, Italy and

\(^8\)https://www.trustpilot.com/
| Rule | Condition | Confidence | \( |X| \) | \( |Y| \) | Lift | Convinction |
|------|-----------|------------|------|------|------|------------|
| r1   | \{memberActivity=1 \& country=fr \Rightarrow geoTransp=low\} | 0.64 | 29,837 | 19,199 | 3.2 | 2.24 |
| r2   | \{gender=male \& country=us \Rightarrow hotelPopularity=very high\} | 0.59 | 44,703 | 26,505 | 1.78 | 1.64 |
| r3   | \{memberActivity=1 \& country=us \Rightarrow guestPics=very high\} | 0.34 | 61,155 | 20,926 | 1.71 | 1.22 |
| r4   | \{memberActivity=1 \& country=us \Rightarrow geoenter=very high\} | 0.54 | 61,155 | 20,4316 | 1.68 | 1.48 |
| r5   | \{memberActivity=1 \& country=us \Rightarrow hotelTripType=very high\} | 0.76 | 44,703 | 34,192 | 1.04 | 1.11 |
| r6   | \{memberActivity=very low \& hotelTripType=very high\} | 0.74 | 27,343 | 20,362 | 1.01 | 1.02 |

Table 7: Excerpt of ARs where user features are premises and the consequences the features of selected hotel, results are sorted by decreasing lift

| Rule | Condition | Confidence | \( |X| \) | \( |Y| \) | Lift | Convinction |
|------|-----------|------------|------|------|------|------------|
| r7   | \{stars=0 \& country=None \& guestPics=very low \& geoTransp=low \Rightarrow ratings=3\} | 0.25 | 9,007 | 2,214 | 1.89 | 1.15 |
| r8   | \{stars=5 \& hotelPopularity=medium \& geoplace=very high \Rightarrow rating=5\} | 0.76 | 2,582 | 1,962 | 1.70 | 2.31 |
| r9   | \{memberActivity=very low \& gender=female \& guestPics=very high \& hotelTripType=very high \Rightarrow rating=5\} | 0.7 | 2,744 | 1918 | 1.57 | 1.84 |
| r10  | \{stars=1 \& country=jp \Rightarrow rating=4\} | 0.47 | 5,265 | 2,492 | 1.38 | 1.25 |
| r11  | \{memberActivity=3 \& guestPics=low \Rightarrow rating=4\} | 0.46 | 4,326 | 1,998 | 1.35 | 1.22 |
| r12  | \{stars=3 \& geoplace=very high \Rightarrow rating=3\} | 0.44 | 4,312 | 1,901 | 1.28 | 1.17 |
| r13  | \{country=jp \& hotelTripType=business \Rightarrow rating=4\} | 0.44 | 4,483 | 1,954 | 1.27 | 1.16 |
| r14  | \{geoenter=very high \Rightarrow rating=5\} | 0.5 | 37,998 | 19,120 | 1.13 | 1.12 |

Table 8: Excerpt of CARs, the class is the review rating, results are sorted by decreasing lift

Fondazione Cariparo, Padua, Italy. The first author would like to thank Giorgio Maria Di Nunzio, for his helpful support.

REFERENCES
Using a NoSQL graph oriented database to store accessible transport routes

Belén Vela
Escuela Técnica Superior de Ingeniería Informática
Rey Juan Carlos University
28933 Móstoles, Spain
belen.vela@urjc.es

José María Cavero
Escuela Técnica Superior de Ingeniería Informática
Rey Juan Carlos University
28933 Móstoles, Spain
josemaria.cavero@urjc.es

Paloma Cáceres
Escuela Técnica Superior de Ingeniería Informática
Rey Juan Carlos University
28933 Móstoles, Spain
paloma.caceres@urjc.es

Almudena Sierra
Escuela Técnica Superior de Ingeniería Informática
Rey Juan Carlos University
28933 Móstoles, Spain
almudena.sierra@urjc.es

Carlos E. Cuesta
Escuela Técnica Superior de Ingeniería Informática
Rey Juan Carlos University
28933 Móstoles, Spain
carlos.cuesta@urjc.es

ABSTRACT

Each day, people have to move to carry out their daily tasks, such as going to work, studying, shopping, etc., signifying that thousands of trips are taken on public transport on a daily basis. A huge number of these trips are taken by people with special mobility needs. In spite of the existence of numerous Websites and apps that provide information about public transport services, there is a lack of information regarding the accessibility of the routes and sites. We are, therefore, working on the development of a technological framework for the processing, management and exploitation of open data, with the goal of promoting accessibility to city public transport within the framework of the Access@city project. In this paper we specifically focus on the design and storage of accessible transport routes, obtained by means of crowdsourcing techniques, in a NoSQL graph oriented database.

1. INTRODUCTION

According to the World Bank [16], one billion people, or 15% of the world’s population, have some type of disability. Although this depends on each country, a significant percentage of the people who use public transport have special mobility needs. One of the goals of smart cities is to improve the quality of life of all citizens [12]. In fact, in a smart city, anyone should be able to move easily and according to their needs. There are, therefore, several initiatives whose objective is to improve accessibility to public transport for people with disabilities. For example, the World Health Organization in its “World Report on Disability 2011” [17] proposes to improve accessibility to public transport for people with disabilities. For example, the website accessible.net shows maps with accessibility information, but does not include search options. The website for disabled people, www.discapnet.es, presents information about training, education, employment, legislation, documentation, organization and related services, and includes guides for accessible transport with the option of searching for routes. There are also websites that provide information regarding accessibility for wheelchair users, such as wheelmap.org and Rollstuhlrouting.de. The abi.io website provides information regarding accessible journeys and service-based routing using public transport. The main reason for this is that there is a significant lack of open and reusable data concerning public transport and its accessibility.

In order to address this lack of open transport data and of information regarding accessibility, we are defining an open data repository for accessible public transport within the framework of the Access@City project. The repository will be developed using a NoSQL database owing to its capacity to manage huge volumes of information, along with its flexibility and scalability [14]. We have specifically selected a NoSQL graph oriented database as we are

The principal eventual aim of our study was to discover the strengths or weaknesses of the public transport information provided and the services offered.

With regard to public transport users, we have analysed the quality and quantity of accessibility information and services; in this case, we have defined six accessibility levels according to the accessibility features related to users’ mobility, visual, audible needs, along with other user needs, and the capacity to provide accessible routes related to those user needs, in addition to assigning an accessibility level to each of the applications studied.

With regard to public transport data, in addition to identifying the accessibility information contained in them, we have also identified their format in order to determine whether the data provided can be simply managed and reused, thus facilitating their extraction from the Internet and their subsequent use.

All of the websites and mobile applications analyzed provide maps and services and some type of accessibility information, but none of them provides generic mechanisms with which to attain accessible transportation data and which would improve mobility in a smart city. For example, the website accessible.net shows maps with accessibility information, but does not include search options. The website for disabled people, www.discapnet.es, presents information about training, education, employment, legislation, documentation, organization and related services, and includes guides for accessible transport with the option of searching for routes. There are also websites that provide information regarding accessibility for wheelchair users, such as wheelmap.org and Rollstuhlrouting.de. The abi.io website provides information regarding accessible journeys and service-based routing using public transport. The main reason for this is that there is a significant lack of open and reusable data concerning public transport and its accessibility.

© 2018 Copyright held by the owner/author(s). Published in the Workshop Proceedings of the EDBT/ICDT 2018 Joint Conference (March 26, 2018, Vienna,Austria) on CEUR-WS.org (ISSN 1613-0073). Distribution of this paper is permitted under the terms of the Creative Commons license CC-by-nc-nd 4.0.
dealing with highly connected data and wish to be able to make queries that are more efficient in a graph oriented database [7].

We propose to develop the graph oriented database, which will be designed from scratch, by using a methodological approach. In general, we have detected a lack of specific methodologies for the design of NoSQL databases that take into account the application characteristics and the most frequent data queries, which is a particularly important aspect in this kind of systems.

A trend concerning how to incorporate traditional modeling notions in this context has recently emerged [2]. For example, in [7], Kaur and Rani show an example of NoSQL database design. They use an Entity Relationship Model [4] to obtain a conceptual representation of the model and different data models for each NoSQL database (for example, a class diagram for a document database). Buggiotti et al. propose a methodology based on an abstract data model for NoSQL databases called NoAM (NoSQL Abstract Model) [3].

In summary, it could be said that different approaches exist but that no solution has, as yet, been commonly accepted. In our opinion, the key concept here is neither the model nor the representation techniques to be used, but rather the design process and the aspects to be considered. The characteristics of NoSQL databases are different in nature from those of SQL databases. Denormalization and queries must be taken into account from the beginning of the process.

In order to address this lack, in our previous work [15], we proposed some guidelines for the design of document databases in which we integrated the final use of the data and the most frequent queries into the design process.

In this paper, we shall show how to design a NoSQL graph oriented database in which to store accessible routes generated by the users of a mobile application. The accessible routes are obtained for users with special needs by using crowdsourcing techniques (micro tasks) [6][8][9]. For the storage of the routes we specifically use what is, according to [13], the most popular graph oriented database i.e., Neo4j [11].

The remainder of the paper is organized as follows: the framework of our work is briefly presented in Section 2. In Section 3 we present our approach for the design of a graph oriented database for the storage of the accessible public transport routes generated. In Section 4 we show a validation of our proposal, along with a brief description of the application developed for the generation of accessible transport routes and their storage in a Neo4j graph oriented database. Finally, our main conclusions and future work are summarized in Section 5.

2. FRAMEWORK

The framework of our paper is the Access@City project, whose objective is to define a technological framework for the processing, management and exploitation of open data with the goal of promoting accessibility to city public transport (see Figure 1). We therefore address the integration of accessibility data derived from three kinds of sources: 1) existing open data, available from Linked Open Data (LOD) initiatives or obtained using the web scraping of non-semantic data sources; 2) private data concerning actual accessible routes, obtained by means of crowdsourcing and provided by users using crowdsourcing techniques and the Big Data repository (REPOSITORY OF ACCESSIBLE ROUTES) that will store the routes. We consider that a route is accessible if a person with a special need can use it to reach his or her destination.

In this paper, we shall show how to design a NoSQL graph oriented database in which to store accessible routes generated by the users of a mobile application. The accessible routes are obtained for users with special needs by using crowdsourcing techniques (micro tasks) [6][8][9]. For the storage of the routes we specifically use what is, according to [13], the most popular graph oriented database i.e., Neo4j [11].

The remainder of the paper is organized as follows: the framework of our work is briefly presented in Section 2. In Section 3 we present our approach for the design of a graph oriented database for the storage of the accessible public transport routes generated. In Section 4 we show a validation of our proposal, along with a brief description of the application developed for the generation of accessible transport routes and their storage in a Neo4j graph oriented database. Finally, our main conclusions and future work are summarized in Section 5.

2. FRAMEWORK

The framework of our work is the Access@City project, whose objective is to define a technological framework for the processing, management and exploitation of open data with the goal of promoting accessibility to city public transport (see Figure 1). We therefore address the integration of accessibility data derived from three kinds of sources: 1) existing open data, available from Linked Open Data (LOD) initiatives or obtained using the web scraping of non-semantic data sources; 2) private data concerning actual accessible routes, obtained by means of crowdsourcing and provided by users themselves through their mobile devices and also processed using Big Data techniques, integrating both historical and real-time data in a datastore [10] denominated as “REPOSITORY OF ACCESSIBLE ROUTES” shown in Figure 1, and 3) data obtained from already existing traffic sensors, in the context of a smart city.

These data sources will be semantically harmonized, while maintaining their diversity, and will feed an open data management platform, which consists of a repository (data hub) and a service generation layer. This layer will be able to provide access to data consumers, through the automatic generation of customized APIs composed of services adapted to available data, which will be exploited by different applications. In particular, we consider the case of mobile applications, which would make it possible for citizens to obtain accessible routes between two points in a city in real time, and even combine different transport networks. These apps would translate the information regarding our smart city into an accessibility context, thus resulting in the definition of an accessible city.

The case study that we present in the following section of this paper is focused on the marked part of Figure 1, which includes the application that captures the accessible routes obtained and validated by users using crowdsourcing techniques and the Big Data repository (REPOSITORY OF ACCESSIBLE ROUTES) that will store the routes. We consider that a route is accessible if a person with a special need can use it to reach his or her destination.

3. DESIGNING A NOSQL GRAPH ORIENTED DATABASE

Our proposal consists of developing the NoSQL graph oriented database by following a process based on the traditional database design. The proposed approach is summarized in Figure 2.

In a first step, we acquire and analyze the data sources or the specification in order to be able to determine the entities and their relationships, along with their properties. This specification is used to define the conceptual data required to design the conceptual schema of the database from scratch. The conceptual model can be represented using, for example, the Entity-Relationship Model [4] or the UML class diagram [13].

In the second step, taking into account the conceptual data model (which is independent of any database technology) and carrying out a study of the application specific access model and the frequent types of queries, we design the logical graph oriented database model, which is independent of any product. This step provides an initial product-independent specification, thus improving the maintainability of the NoSQL database, in addition to making migrations between products easier.

In the third step, we attain the physical design and the implementation for a specific NoSQL product, and the product database model is obtained. In our case, we have chosen Neo4j [11], which is, according to the database ranking [5], the most popular graph oriented database. Finally, the implementation phase includes
various physical design tasks, such as balancing the need for scalability, availability, consistency, partition protection and durability.

Figure 2. Graph Oriented Database Design Approach

Focusing on the second step, that is, on the transformation of the Conceptual Data Model into the Logical Graph Oriented Model (red arrow in Figure 2), we shall begin with a conceptual model represented using the Entity-Relationship (E/R) Model. For the logical graph oriented model, we shall consider “directed graphs”, which are graphs composed of nodes (or “vertices”) connected by relationships called “edges”, each of which is associated with a direction. The direction of the edges is represented by means of an arrowhead on the connecting line between the nodes.

With regard to the transformation, we consider the E/R schema obtained in the first step, the most common queries as regards the data (defined in natural language) and the update operations performed in the database by the applications in an iterative process.

Bearing these aspects in mind, along with the fact that the data can be queried in many ways, we have to decide when to transform an entity or a relationship into a node type or an edge.

In a first iteration, the summarized rules are:

- Each entity will be transformed into a node type labelled as the entity and its attributes into properties of this node type.
- Each relationship will, in general, be transformed into an edge between the nodes, depending specifically on the cardinality of the relationship.
  - One-to-one relationships (0/1 to 0/1) will be transformed into an edge (without an arrowhead) between both node types to denote the 1:1 relation between the entities.
  - One-to-many relationships (0/1 to 0/n) will be transformed into an edge with an arrowhead to denote the 1:N relation between the entities.
- Many-to-many relationships (0/n to 0/m) will be transformed into an edge between both node types with an arrowhead on each end to denote the N:M relation between the entities. At this point, we have to decide and check whether this relationship should be transformed into an edge or a node.
- A generalization is a special kind of relationship and will be transformed in the same way as the other types of relationships, according to its cardinality and including an edge labelled “is-a”.
- A composition is a special kind of relationship and will be transformed in the same way as the other types of 1:N relationships, according to its cardinality and including an edge labelled “is-composed-of”.

After this first iteration, we have to refine our logical graph oriented DB model, taking into account both the access patterns of the applications and frequent queries in order to be able to query the connected data in many ways, as required by the users.

4. VALIDATION: APPLICATION FOR GENERATING AND STORING ACCESSIBLE ROUTES USING A GRAPH ORIENTED DATABASE

In order to validate our proposal, we have developed a native Android application as we need to use the GPS of the users’ device. In general, native applications have significant advantages over hybrid applications because they are able to easily access and use the built-in capabilities of the user’s devices (e.g., GPS) [1].

This application will allow users to register for the generation of accessible routes. They can then use the starting a route option, indicating which special need (wheelchair, bike, baby stroller, baby buggy, etc…) they will have on their journey. During the journey, the application will periodically register the GPS position (initially, every 25 seconds, although this could change depending on the route, the special need, etc.). When the user finishes the journey, he/she can either discard the route or save it. Users may include comments about the routes taken, reporting possible incidents and/or including photos.

Figure 3 shows the main functionalities of the application by means of a Use Case Diagram:
For the storage of the routes, we have developed a Big Data repository in which to store accessible routes obtained by means of the aforementioned application using crowdsourcing techniques. The Big Data repository in our Case Study was developed by first identifying the data requirements. We then defined the conceptual data model using an Entity-Relationship Data Model (Figure 5).

At this point, another necessary and important decision that had to be made was which kind of NoSQL database to use for the development of our big data repository. In this work, we have chosen a graph oriented database owing to the nature of the data of the routes, which is highly connected, and the need to query the data in many ways. The methodology shown in Figure 2 assumes that the database chosen is a NoSQL graph oriented database. The case study has been implemented in Neo4j. The application will be connected to the Neo4j database using API REST and an HTTP connection.

In a previous work [15], we developed our repository using a NoSQL document database because of its flexibility and ability to manage complex data structures [14]. However, as the data in our case study are highly connected (point of a route), the traversal is much simpler using a graph oriented database.

In order to obtain the logical graph oriented database model, in a first iteration we have to apply the proposed guidelines. We then have to consider how the data will be used by the applications, that is, what the most frequent queries and application-specific access models are.

In our case study, the most common queries will be related to users or to routes. The most frequent queries are:

1) Data of the registered users, including their default special need.
2) Is a route between two given points accessible?
3) Comments concerning or photos of the stored accessible routes.
4) Of which points is a route composed?

Apart from the queries, there will also be two basic update operations in the database: inserting a new registered user or inserting a new route.

Bearing in mind the conceptual data model and the aforementioned queries (defined in natural language), we have designed the new model (according to our proposed guidelines):

a) A node type labelled for each entity: User, Special Need, Route and Point. The attributes of the entity become the properties of each node type, as can be seen in Figure 6.

b) It is now necessary to carry out the transformation of the existing relationships. Here, we have to decide whether relationships will be implemented using an edge or a node.
There are three relationships: Has Default relationship (1:N), Create relationship (1:N:M) and Contain relationship (1:N).

c) When analyzing the queries and the application data, we consider grouping the information of certain entities of the conceptual data model that will be used together. We therefore exclude the Special Need node, as its information can be considered as information regarding the User and the Route. We shall, therefore, include the Special Need information as properties of the User and Route node types. This signifies that we now continue working with only three node types and two relationships.

In order to transform both 1:N relationships, we create an edge labelled as the relationship with an arrowhead to denote the 1:N relation between the entities: Create and Contain.

Figure 7. Node types with edges

The final graph oriented database design will, therefore, consist of three node types: USER, ROUTE and POINTs. USERS will store, for each user, a set of information (name, email, birth date, etc.) and information on their default special needs. ROUTES will store the special needs for that route, and some additional information, such as comments, photos, etc. POINTs will store information concerning the X and Y coordinates and the type of point (initial, intermediate and end point). The information regarding routes and subroutes with a special need can be easily obtained with this design.

5. CONCLUSIONS

Although document databases have a dynamic schema (but are not schemaless, as stated in some forums), it is very important to design this schema correctly because of its impact on the performance of the database. A methodological process with which to guide the user in the design process of a NoSQL database is, therefore, required. However, despite the existence of several works and many websites with best practices, there is no commonly accepted solution for the design of NoSQL databases.

In [15], we proposed some guidelines for the design of a document database. In this paper, and in order to complete our NoSQL Design Methodology, we address the design of NoSQL graph oriented databases. In order to validate this proposal, we present a case study in which we generate accessible routes created by users with special mobility needs using a micro-task based on crowdsourcing and store them in a NoSQL graph oriented database in Neo4j. The design process is principally based on the requirements of the applications and the most frequent queries that the system will have to deal with.

One of our future works will be the formal representation of the queries and the specification of a formal method with which to transform the conceptual model and the query model into a logical design model based on a NoSQL database (document DB or graph oriented DB). We plan to put the application into use with users with disabilities in the near future. Moreover, an immediate future task is to extend the functionalities of the mobile application in order to create a version that can be evaluated by users with special needs.

ACKNOWLEDGMENTS

This work was partially supported by the Access@City project (TIN2016-78103-C2-1-R), funded by the Spanish Ministry of Economy and Competitiveness.

REFERENCES


Guided Query Composition with Semantic OLAP Patterns

Ilko Kovacic
Johannes Kepler University Linz
Linz, Austria
ilko.kovacic@jku.at

Christoph G. Schuetz
Johannes Kepler University Linz
Linz, Austria
christoph.schuetz@jku.at

Simon Schausberger
Johannes Kepler University Linz
Linz, Austria
simon.schausberger@jku.at

Roman Sumereder
Johannes Kepler University Linz
Linz, Austria
roman.sumereder@jku.at

Michael Schrefl
Johannes Kepler University Linz
Linz, Austria
michael.schrefl@jku.at

ABSTRACT

Enabling domain experts to independently compose ad hoc OLAP queries is the primary goal of semantic OLAP (semOLAP) patterns. In this respect, a semOLAP pattern represents a recurring domain-independent OLAP query by describing the application scope and defining the structure of the query using formal pattern elements (FPEs). Such a semOLAP pattern is executable: in order to execute a semOLAP pattern, the user instantiates the pattern by providing FPE bindings. In this paper, we propose an approach for guided query composition which considers the inherent query structure in order to determine a navigation flow and recommend possible bindings for the corresponding FPEs. Guidance supports both existing as well as future, currently unidentified semOLAP patterns. The presented approach has been implemented in the course of a collaborative research project between industry and academia on precision dairy farming.

1 INTRODUCTION

Data warehousing and online analytical processing (OLAP) facilitate data-driven decision making, allowing domain experts to make rational decisions. A data warehouse organizes data in a multidimensional space (data cube). Each point in such a multidimensional space represents an occurrence of a business event (fact) which is quantified by measures. Hierarchically organized dimensions support the aggregation of facts along a hierarchy of granularity levels, e.g., day to month, city to county.

Standardized reports provide access to data warehouses in order to satisfy the domain expert’s information needs. These reports are usually not static but rather support the specification of selection criteria restricting only one dimension (slice) or multiple dimensions (dice). Each report executes a predefined underlying query – an OLAP query – to retrieve the required information. Reports, however, can only satisfy about 60-80% of the information needs [6, p. 19]. Satisfying the remaining information needs requires the composition of ad hoc OLAP queries.

In order to compose ad hoc OLAP queries, domain experts must have knowledge about the underlying schema and the employed query language. Domain experts, however, typically lack the required knowledge and, therefore, must rely on assistance for ad hoc OLAP query composition.

This research was conducted as part of the agriProKnow project (http://www.agriProKnow.com/), funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program “Production of the Future” between 11/2015 and 01/2018, Grant No. 848610.

© 2018 Copyright held by the owner/authors. Published in the Workshop Proceedings of the EDBT/ICDT 2018 Joint Conference (March 26, 2018, Vienna, Austria) on CEUR-WS.org (ISSN 1613-0073). Distribution of this paper is permitted under the terms of the Creative Commons license CC-by-nc-nd 4.0.
In this paper we propose a guidance approach for semOLAP pattern instantiation. After selection of a semOLAP pattern, the user is guided through the steps of the instantiation process following a navigation flow. This navigation flow connects all activities required to instantiate the FPEs. For each FPE, possible bindings are recommended to the user. To this end, the guidance approach relies on a semOLAP knowledge graph, which contains knowledge about the pattern structure, the underlying schema, and the bindings of already instantiated FPEs. The semOLAP knowledge graph enables the recommendation of bindings which are suitable for specific FPEs. The user can select values as bindings during the instantiation of the pattern without deeper knowledge of the underlying schema, dependencies between query elements, and the query language. After all FPE bindings are specified by the user, the instantiation process is finished and a corresponding query is generated in order to retrieve the required information. The implementation of the data warehouse employs a relational database where different fact classes are stored as fact tables. The semOLAP knowledge graph is represented in Resource Description Framework (RDF) format. The interaction flow modelling language (IFML)\(^1\) and the WebRatio\(^2\) platform are used for the implementation supporting a model-driven and data-centric development.

The remainder of this paper is organized as follows. Section 2 discusses the semOLAP approach. Section 3 details the semOLAP knowledge graph. Section 4 explains the determination of the navigation flow and the recommendation of possible bindings. Section 5 exemplifies the instantiation of a semOLAP pattern. Section 6 reviews related work. The paper concludes with a summary and an outlook on future work.

## 2 SEMANTIC OLAP PATTERNS

The notion of patterns is introduced by Alexander et al. [1] where patterns describe how a specific problem in a specific context can be solved while considering existing constraints. In OLAP, an unsatisfied information need represents the problem whereas the specific analysis situation represents the context [10]. A semOLAP pattern can, therefore, be seen as an instruction on how to compose an OLAP query that satisfies the information need in a specific analysis situation. The identification of such patterns is based on the detection of recurring OLAP queries, which are usually abstracted to domain-dependent templates for OLAP reports. To obtain domain-independent semOLAP patterns, such templates are grouped and abstracted (see Fig. 2).

As of now, the identified patterns can be grouped into basic patterns and comparative patterns. The group of basic patterns covers generalized multidimensional queries aggregating business events (facts) according to spatial, temporal, and/or semantic aspects. Domain experts perform such a query by joining one fact class with its dimensions, restricting the result of the join using selection criteria (business terms), grouping the result using grouping criteria, and aggregating measures by applying predefined aggregation functions (calculated measures).

In contrast to basic patterns, which are only based on one set, comparative patterns serve to compare two sets. Therefore, a set of interest (SI) and a set of comparison (SC) need to be defined. The SI is used to specify the primarily focused data which is compared to another set, the SC. For each of these two sets, either the same or different fact class(es), selection criteria, dimension(s), grouping criteria, and/or measure(s) are defined. Depending on the number of shared pattern elements, different types of comparative patterns can be identified. The homogeneous set-base comparison pattern, for example, covers all OLAP queries where a subset (SI) is compared with its base set (SC). It is a homogeneous comparison because both SI and SC refer to the same fact class. The grouping criteria, measures, and selection criteria are shared, with the exception of additional selection criteria which are exclusively used to define the SI. The heterogeneous independent set comparison pattern, contrary to the previously described patterns, is not restricted to one fact class. It is heterogeneous since two different fact classes are used to define SI and SC. Furthermore, no pattern elements at all must be shared. This also applies to the measures to be compared since they can be based on completely different aggregation functions. The measures from SI and SC can be used to calculate ratios, rates, percentages, proportions, and other complex values.

The definition of such semOLAP patterns is based on semantic web technologies, i.e., RDF, yielding formalized and machine-readable representations. Furthermore, RDF allows to define shared conceptualizations representing calculated measures and business terms (predicates) which can be used during pattern instantiation and linked to domain ontologies. Each pattern definition comprises a textual description, a target language-dependent pattern expression, the pattern result, and the FPEs defining its structure.

As the target audience are domain experts the textual description includes all relevant information needed to instantiate the pattern. Therefore, each semOLAP pattern definition covers a concise pattern name, a description of the analysis situation where it can be applied in, the solution describing the instructions to follow, and an example. In addition to the textual description, a pattern definition contains a pattern expression. This pattern expression is a representation of the query to be generated in a specific target language, e.g., SQL. This representation is enriched by grammar expressions which indicate where certain FPE values must be placed in order to generate an executable query. The result of a pattern is specified by defining which FPEs are returned, i.e., which measures and grouping criteria are returned and how they can be enriched by prefixes to foster differentiation of set-specific elements. It is specified only once in the pattern

---


---

![Figure 1: Pattern instantiation guidance activities](image1)

![Figure 2: Query abstraction levels](image2)
definition and not changed during the instantiation. Reusability is fostered, since each result yields a new cube which again can be used as the fact class in other pattern instances.

The structure of an OLAP query is represented by FPEs, which are defined as objects in the pattern definition but treated as properties during the instantiation. The FPE siFactClass, for example, is used to define the fact class of the SI during the pattern instantiation of the heterogeneous independent set comparison pattern. To support such a behaviour an FPE consists of an element range, a multiplicity, and is part of zero or more pattern element sets (see Fig. 3). The FPE range defines the (sub)type of the values which can be specified during the instantiation. For the FPE siFactClass, for example, the range is set to the Fact type. Depending on the FPE, the range can be set to (sub)types representing measures, dimensions, dimension attributes, and predicates. The multiplicity, as the name suggests, determines the number of values that can be provided for an FPE during the pattern instantiation, such as One or OneOrMore. The FPE siFactClass, for example, is defined with the multiplicity One specifying that only one value of the Fact type can be used for the definition of the SI. As already indicated by the prefix of the name siFactClass, FPEs can be assigned to pattern element sets, e.g., SC or SI, using the partOfSet property. This is especially important during the instantiation of SI and SC, since different selection criteria can be applied to different pattern element sets.

Figure 3: Formal pattern element structure

FPEs can also be related to each other using the dependsOn property. The fact class, for example, which stores occurrences of a business event, is the core element of the multidimensional model. These stored occurrences are quantified by measures, hence, there exists a dependency between a measure and its corresponding fact class. Further dependencies exist, since a fact class can be aggregated to different levels of granularities according to its corresponding dimensions and hierarchies. Each fact class has predefined dimensions and each dimension can be assigned also to different fact classes. Dimensions support the aggregation of fact classes to different levels of granularity and, therefore, each dimension has one or more dimension hierarchies which, again, consist of dimension attributes. All these dependencies between FPEs are expressed by dependsOn relationships. In addition to the dependencies within the pattern element sets SI and SC, dependencies of FPEs located outside of the pattern element sets can exist. Comparative measures, for example, are defined by using measures from both SI and SC. Comparing two element sets can exist. Comparative measures, for example, are defined by using measures from both SI and SC. Comparing two element sets can be provided for an FPE during the pattern instantiation, such as One or OneOrMore. The FPE siFactClass, for example, is defined with the multiplicity One specifying that only one value of the Fact type can be used for the definition of the SI. As already indicated by the prefix of the name siFactClass, FPEs can be assigned to pattern element sets, e.g., SC or SI, using the partOfSet property. This is especially important during the instantiation of SI and SC, since different selection criteria can be applied to different pattern element sets.

Figure 3: Formal pattern element structure

A typical OLAP query is composed of the fact class representing the data of interest, grouping criteria, and selection criteria representing logical restrictions regarding temporal, spatial, and semantic aspects. The semOLAP pattern definitions reflect this structure by specifying FPEs and the relationships between them, e.g., the set of selectable measures and dimensions depends on the previously selected fact class. The pattern definition also includes constraints for each FPE: multiplicity, element range, and the pattern element sets to which the FPE is related. This available knowledge, also called pattern knowledge, can be exploited during pattern instantiation, e.g., to determine the FPE instantiation order or the type of possible values for an FPE. The pattern knowledge graph in Fig. 4 is an extract of the FPE’s dependsOn relationships of an SI definition. The siSlice depends on the siFactClass and the siDimension, whereas the siDimension depends only on the siFactClass. The ranges for these FPEs are represented by the (sub)types of their values, e.g., for the siFactClass the FPE range is the type Fact.

The types of the FPE ranges are part of the underlying semantic schema knowledge. The schema is based on the Dimensional Fact Model (DFM) [8] which allows to conceptually represent multidimensional elements such as fact classes, attributes, dimensions, dimension hierarchies, and relationships between them.
The modeled elements are represented using the RDF Data Cube (QBO) [5] vocabulary and its extension QB4OLAP [7], thus creating a semantic multidimensional schema. This RDF representation facilitates the definition of predicates (ObjectPredicates) representing business terms as well as calculated measures (CalculatedMeasures) which can exceed simple aggregations. The semantic schema knowledge graph in Fig 4 shows the types and subtypes of the range of the FPEs and the structural relationships (dotted directed requires edge) between these (subtype)FPEs. During the instantiation process this RDF knowledge provides information about the structure of the type, e.g., the type ObjectPredicate and some of its subtypes require the structure provided by (sub)types of the ranges of siFactClass and/or siDimension.

In addition to the pattern and the semantic schema knowledge, the binding knowledge has to be considered. It represents the current instantiation, i.e., the bindings of FPEs within the instantiation process. The pattern instance, again represented in RDF, is updated during the instantiation process. The binding knowledge contains the already instantiated FPEs with their values and all currently uninstantiated FPEs. During the binding recommendation process, the binding knowledge needs to be considered, since it reflects the available structure of existing values on the basis of which suitable values can be determined. The current binding knowledge graph in Fig 4 depicts the available fact class value BCS and the dimension level value MainBreed for siFactClass and siDimension. These values must be considered to recommend values for siSlice, e.g., in order to recommend the FactDimensionPredicate value riskOfObesity, it is checked if its structurally required values BCS and MainBreed exist in the binding knowledge.

A guidance approach for query instantiation requires to consider the whole knowledge graph in order to provide navigation and recommendation and to avoid the creation of invalid queries. Valid queries can be only ensured when all relationships between the pattern to be instantiated, the semantic schema elements, and the already provided values are considered.

4 EXPLOITING SCHEMA AND PATTERN KNOWLEDGE FOR INCREMENTAL PATTERN INSTANTIATION

The semOLAP patterns provide a conceptual foundation to compose ad hoc OLAP queries without further assistance. A domain expert, however, requires visual assistance to fulfill this task. They should be enabled to browse existing semOLAP patterns, select the one which fits their information need, and instantiate the semOLAP pattern in order to generate the desired query. Especially the pattern instantiation is a non trivial task since the available knowledge graph, which can be used to determine and restrict possible values for FPEs, must be considered (see Fig. 1).

The guidance process based on semOLAP patterns requires the consideration of the semOLAP knowledge graph as well as an interactive instantiation interface. The interface implementation is based on IFML which supports a data-driven application development following a strict separation of the data model, the hypertext model, and the presentation model [4]. We focus on the hypertext and presentation model since these are crucial for the user interaction. Furthermore, interfaces are generated for the browsing, selection, instantiation, and result retrieval step. To detail the guidance approach and the implementation, the instantiation of the heterogeneous independent set comparison pattern is exemplified (see Fig. 5).

4.1 Navigation Flows

Adapting the idea of logical stratification [12, p. 131-136], we determine a default navigation flow by calculating the corresponding level of each FPE. The calculation of the levels is based on the FPEs from the pattern knowledge and simple rules: FPEs with no outgoing dependsOn edge are assigned to level 0; FPEs which have one or more outgoing dependsOn edges are assigned to the highest level of the referred FPEs plus one; these steps are repeated until the level assignments are not changed any more (see Algorithm 1).

```plaintext
Algorithm 1: Level computation
repeat
  forall formalPatternElement fpe in patternKnowledgeGraph do
    level[fpe] := 0;
  end

repeat
  forall formalPatternElement fpe in patternKnowledgeGraph do
    forall dependsOn dp in fpe.dependingOn do
      targetFpe := dp.target;
      if level[fpe] <= level[targetFpe] then
        level[fpe] := level[targetFpe]+1;
    end
  end
until there are no changes to any level or a level exceeds the number of formal pattern elements;
until all levels of abstraction are processed;
```

An exemplified application of this algorithm is the calculation of the SI levels depicted in Fig. 5. The calculated level assignments are indicated by the number in the left corner of the instantiated FPEs. The first number indicates the assigned level whereas the second number indicates the sequence within the default navigation flow. The siFactClass is assigned to level 0 since it has no outgoing dependsOn edges; siMeasure and siDimension are assigned to level 1 due to their dependence on siFactClass; siDimension.Attribute and the siSlice are assigned to level 2 due to their dependence on siDimension. This algorithm, however, is not limited to the FPEs in the pattern element sets SI and SC, it can also be applied to the next level of abstraction. Each pattern element set can be also seen as an FPE of an outer graph. Considering this abstraction level both SI and SC represent FPEs without outgoing dependsOn edges which are referenced by the FPEs factCorrelation or the compMeasure.

The default navigation flow can be determined by considering the dependsOn relationships between the FPEs and the assigned levels. It starts from FPEs assigned to the lowest level, i.e., from FPEs assigned to level 0. If multiple FPEs are assigned to the same level an arbitrary navigation flow order can be specified for them. These steps are repeated for the next levels until all levels are processed. The result is a default navigation flow linking the interface elements of the FPEs during the instantiation of a selected pattern.

The dotted directed links to edges in Fig. 5 represent the default navigation flow of the heterogeneous independent set comparison pattern. It starts with the value specification of siFactClass, followed by the specification of siMeasures and siDimension. The order of these last two FPEs is interchangeable since both of them
are assigned to level 1. At level 2, values must be specified for the dimension attributes siDimensionAttribute, representing the grouping criteria, and the siSlice, representing set-specific predicates. The selectable dimension attributes depend on the specified dimensions. Depending on the values specified in the previous levels, only specific types of predicates can be specified for the siSlice. In the heterogeneous independent set comparison pattern, the FPE dependsOn relations of SI and SC are the same, therefore SI’s navigation flow can be applied to the SC analogously. To finish the instantiation of the pattern the factCorrelation attribute, used for joining, and the compMeasure, which determines the type of comparison to be performed, have to be specified.

The default navigation flow only allows slight adaptations, such as changing the FPE order within one level, e.g., either siMeasure or siDimension can be instantiated first. Additional adaptations, however, have to be supported since a user might not want to start with the specification of the FPE siFactClass, instead they might want to start with other FPEs. As discussed in [3], a user knows prior to the query composition which measure(s) they want to retrieve, therefore a user typically starts with the selection of the desired measure(s). This is especially relevant for ad hoc query composition, since a user wants to retrieve something that is not covered by existing reports. Facilitating a custom navigation flow requires the adaptation of the default navigation flow which is based, so far, on the FPEs’ dependsOn relationships and the assigned levels. The navigation flow must be detached from these dependsOn relationships to provide such flexibility. A custom navigation flow cannot be determined automatically, instead it must be specified manually in the course of system configuration. In exchange, the custom navigation flow allows to move arbitrarily between the FPEs, e.g., allowing to navigate from siMeasure to siFactClass.

4.2 Binding Recommendation

The navigation flow allows to move from one interface element to another while providing values for the corresponding FPEs. The user can be guided in this process by having bindings recommended for the FPE values. Therefore, the range of the current FPE (available in the semantic schema knowledge), the dependsOn relationships between FPEs (pattern knowledge), and the bindings of other FPEs (binding knowledge) need to be considered. To illustrate this, we exemplify the instantiation of the FPE siSlice in the heterogeneous independent set comparison pattern by recommending bindings of the FactDimensionPredicate subtype; this approach can be applied to all other FPEs and (sub)types as well.

Each FPE specified in the pattern definition is represented as a property during the pattern instantiation. The range of each FPE determines the type of possible bindings, e.g., the range of siSlice is ObjectPredicate. Due to the complexity of the multidimensional model, each range can cover multiple subtypes which are a part of the semantic schema knowledge, e.g., ObjectPredicate is a subtype of Predicate and a supertype of FactPredicate, DimensionPredicate, and FactDimensionPredicate. Consequently it is possible to select bindings of the types FactPredicate, DimensionPredicate, or FactDimensionPredicate for the FPE siSlice (see pattern knowledge and the semantic schema knowledge graph in Fig. 4). We focus here on bindings of the type FactDimensionPredicate.

Recommending bindings for the current FPE requires to retrieve all other FPEs that the current FPE depends on – the depending FPEs. To this end, dependsOn relationships from the pattern knowledge graph are used. Considering, for example, the dependsOn relationships of the siSlice allows to identify the depending FPEs siFactClass and siDimension. For each subtype of the current FPE’s range, each depending FPE is processed separately. For the sake of simplicity we refer to the (sub)type of the current FPE’s range as current subtype and to the subtypes of the depending FPEs’ range as depending subtypes. For example, for the current subtype FactDimensionPredicate, as a subclass of siSlice’s range, the depending FPE siDimension is processed. Therefore, the (sub)types of the depending FPE’s range are retrieved. For the range of the depending FPE siDimension, for example, the subtypes are DimensionLevel and DimensionRole.

For the current subtype FactDimensionPredicate and the depending subtypes DimensionLevel and DimensionRole the possible basic relations FactDimensionPredicateRelatesToDimensionLevel and FactDimensionPredicateRelatesToDimensionRole need to be considered. A basic relation is used to represent the structural relationship of a current subtype to a depending subtype (see requires relationships in the semantic schema knowledge graph in Fig. 4). Not all possible basic relations derived from current and depending subtypes actually exist. The siSlice, for example, depends on siDimension but the FactPredicate, which is a subtype of siSlice’s range, does not have a basic relation to either subtypes of siDimension’s range DimensionLevel nor DimensionRole. Therefore, only the actually existing basic relations are then used

A dimension role is used to reference dimensions using different names, e.g., the dimension animal can be references using the dimension role dam animal.
to determine potential bindings of the current subtype for the current FPE. Each of the existing basic relations is represented by a predefined SPARQL Protocol And RDF Query Language (SPARQL) query which checks all available values of the current subtype in order to determine potential bindings.

The bindings of the depending FPE that are of the depending subtype are checked against the required structure of each available value of the current subtype. The structure of values is represented by relationships in the multidimensional schema, predicates, and measures, e.g., the FactDimensionPredicate riskOfObesity requires the dimension attribute MainBreed and the measure BCS (see Fig. 4). If the required structure regarding the current basic relation is available in the structure of the depending FPE binding, the value of the current subtype is added to the list of potential bindings of the current depending FPE. Determining if riskOfObesity, for example, is a potential binding for the current FPE, the underlying SPARQL query of the basic relation FactDimensionPredicateRelatesToObjectCalculatedMeasureRelates to bidirectional ones. After the dependsOn relationships are extended, the corresponding basic relations have to be defined. Therefore, the basic relations of the subtypes of the FPE’s ranges need to be extended by relations in the opposite direction, e.g., for the FPEs siSlice and siFactClass with their corresponding ranges ObjectPredicate and Fact the existing basic relations are extended by FactRelatesToObjectCalculatedMeasureRelates, FactRelatesToObjectDimensionPredicate, and FactRelatesToObjectDimensionLevel.

If a user, for example, starts to select a binding for siMeasure and then navigates to the FPE siFactClass, the basic relation FactRelatesToObjectCalculatedMeasureRelates can be used to determine fact values, which provide the necessary structure for the previously specified siMeasure value(s). This, again, takes the semOLAP knowledge graph into account. The binding recommendations for a custom navigation flow, however, faces also limitations. For example, the instantiation could start with the specification of the siMeasure value, followed by the siDimension value and continuing with the siFactClass value. Recommending possible bindings for siDimension would not be possible, since no basic relations between the subtypes of siMeasure and siDimension ranges exist in the heterogeneous independent set comparison pattern. All potential bindings, in this case all values of the subtypes DimensionRole and DimensionLevel, would be recommended as possible bindings. This issue can be solved by introducing new basic relations between the subtypes of siMeasure and siDimension. These new basic relations, which do not follow existing dependsOn relationships, can be created by considering existing basic relations between subtypes of siMeasure and siFactClass and subtypes of siFactClass and siDimension. In contrast to siDimension, possible bindings could be recommended for the siFactClass without new basic relations, since there exist dependsOn relationships between the siFactClass and both siDimension and siMeasure. The bidirectional basic relations FactRelatesToObjectDimensionRole, FactRelatesToObjectDimensionLevel, and FactRelatesToObjectCalculatedMeasure are therefore used. Considering these relations allows to recommend bindings for the instantiation of the FPE siFactClass, however, it could be possible that no bindings at all could be recommended. This would be the case if, for example, a combination of values for siMeasure and siDimension is selected which cannot be structurally supported by any value of siFactClass. To resolve this issue the user would need to navigate back and edit the corresponding FPE values, otherwise the instantiation could not be continued. The default navigation flow is not affected by these limitations at all, since it considers the logical dependencies derived from the multidimensional model.

Binding recommendations are restricted by the availability of existing bindings of the depending FPEs. The advantage of this approach is that the basic relations are defined only once and can be reused multiple times since FPEs with the same dependencies are used within different pattern definitions. For example, the basic relation between the subtype FactDimensionPredicate and DimensionLevel occurs in the basic multi-aggregation patterns as well as in other comparison patterns such as the homogeneous set-base comparison pattern. New FPEs with currently not considered dependsOn relationships can be introduced as part of new semOLAP patterns. To handle these dependencies only their basic relations and the underlying SPARQL queries need to be defined once. Contrary to this case, new semOLAP patterns using only considered dependsOn relationships, can be instantiated without further effort.
5 EXEMPLIFIED PATTERN INSTANTIATION

The guidance approach is exemplified by instantiating the heterogeneous independent set comparison pattern following the default navigation flow. Even though custom navigation flows could be supported, only the unidirectional basic relations are implemented so far. To illustrate our approach we consider a domain expert who wants to compose an ad hoc query which calculates the ratio of the consumed food of one day (SI) and the milk yield of the next day (SC) for the same animal in June 2016 per date and animal (see FPE bindings in Fig. 5). The resulting ratio is used by the expert to see whether the amount of food fed the day before impacts the milk yield of the next day. Therefore, a data cube containing the two fact classes Milk and Feeding with the shared dimensions Date, Farm Site, and Animal is accessed (see DFM model in Fig. 6).

The instantiation starts with the specification of the SI and its corresponding FPEs. The available fact classes Milk and Feeding are recommended, besides others, to the domain expert. The domain expert selects Feeding as the binding value for the siFactClass FPE. The specified fact class Feeding and the corresponding basic relations are used to determine possible bindings for siDimension, i.e., the dimensions Animal, Date, and Farm Site. The domain expert selects the Animal and Date dimensions as values for siDimension and continues with the specification of the measure of interest siMeasure, i.e., the consumedWeighedRough representing the summarized values of consumed weighed roughage. For the siDimensionAttribute the dimension attributes Animal and Date binding values are selected. The names of the values of siDimension and siDimensionAttribute are the same, even though they represent different FPEs. This is the result of naming conventions, since the dimension’s name is used as the name of the identifying dimension attribute (see Fig. 6). Based on the dimension values Animal and Date and the fact value Feeding, DimensionPredicate values, such as June2016 or MainBreedHolstein, FactPredicate values, such as HighFoodConsumption, and FactDimensionPredicate values, such as HolsteinWithHighFoodConsumption, are recommended to the domain expert as possible values for siSlice (see recommended bindings in Fig. 5). To finish the instantiation of the SI, the domain expert selects June2016 as the binding for siSlice.

The instantiation of SC starts after the instantiation of SI is finished. Since the default navigation graph of SC and SI are isomorphic, the domain expert is guided through the same steps. The domain expert provides the same value bindings to scDimension, scDimensionAttribute, and scSlice already used for the analogous FPEs of the SI. This is possible since in this query SI and SC share the dimensions, dimension attributes, and slices. Only for the FPEs scFactClass and scMeasure different bindings have to be specified. The Milk fact is specified as the binding of scFactClass and SumOfMilkYieldParlour measure as the binding of scMeasure.

After both sets are instantiated, the domain expert specifies the bindings matchDayToDateAfter and sameAnimal for the factCorrelation and FoodMilkYieldRatio for the compMeasure to finish the instantiation process. Similar to the previous instantiation steps, other possible values for these two FPEs are provided, however, only the mentioned are relevant for the intended query. The factCorrelation value matchDayToDateAfter combines the fact occurrences of SI of one specific day with the fact occurrences of SC of the next day whereas sameAnimal restricts the combinations to the same animals. The FoodMilkYieldRatio calculates the ratio between SI’s consumedWeighedRough and SC’s SumOfMilkYieldParlour. After all FPE values are specified, a summary of the pattern instance is provided (see Fig. 7). This summary provides an overview of all FPEs of a semiOLAP pattern instance and indicates which FPEs are differently specified in SI and SC. The differences are color-encoded to ease their identification. The structure of the overview is independent of the semiOLAP pattern but it can be adapted by the developer to consider characteristics of certain patterns.

Finally, to satisfy the domain expert’s information need, the OLAP query is generated and sent to the underlying ROLAP system. The result of this OLAP query is visualized and can thereby be interpreted by the domain expert. The domain expert can reuse this pattern instance for future analysis situations and adapt it to fit their information need. Besides fully-instantiated patterns, partial instances can be specified as well, e.g., the fact, dimension, dimension attributes, and measures are predefined and only additional selection criteria can be specified. These partially-instantiated patterns can be reused as domain-dependent query templates. A video4 of this instantiation process is provided to demonstrate the current state of the implementation.

6 RELATED WORK

The guidance of users during the OLAP query composition is mostly accompanied by providing suitable visualizations of the query elements. The Semantic Data Warehouse Model (SDWM) allows for a visual specification of multidimensional queries. The SDWM does not represent semantic representations of the multidi-dimensional data model, e.g., using QB and QB4OLAP, instead the SDWM considers both the operational requirements as well as the semantics of the business processes to be modeled [3]. Therefore, templates which are based on the SDWM are provided to users, serving as configurable reports [3]. Each of these templates consists of predefined measures, dimensions, and dimension attributes which are related to each other. The relationships between these template elements are visualized to represent the dependencies between the measures and dimensions. The user specifies the OLAP query by either adding/removing new measures or by selecting the dimension hierarchy levels. Similar to our approach, only possible dimension and dimension

---

4 https://www.youtube.com/watch?v=BLt6heO7WKY
attribute values are provided to the user, however, this is not ensured through reusable basic relations. Furthermore, additivity checks are performed, which restrict aggregations to only possible measures, e.g., it does not make sense to summarize the food to milk ratio over time. The proposed approach using the SDWM, however, does not provide the abstraction of semOLAP pattern, since it focuses on case-specific templates which are restricted to a corresponding fact class. The user is not able to specify fact values nor complex predicate values; only simple restrictions are supported. Furthermore, the composition of ad hoc queries targeting multiple fact classes is not considered at all.

Another query visualization interface is Polaris [11] which led to the development of Tableau®. Polaris focuses on analyzing, querying, and visualizing multidimensional relational databases although newer versions of Tableau support other data structures as well. Instead of focusing on ad hoc queries, it primarily supports the explorative data analysis approach by providing an interactive visualization of both the query and the result. The query is defined by a visual specification within a table-based interface, which allows to specify dimensions, measures, and grouping and filter criteria along with possible visualisation options. Corresponding queries are generated using an underlying table algebra. Ad hoc queries can be formulated since all functionalities for composition are provided, however, analysis situation-specific guidance is not supported. The user is not guided while creating, for example, comparisons of sets from one or multiple fact classes, even though, the necessary functionality is available. The application of filters is provided, however, these are restricted to simple expressions; predicates representing business terms are not supported. Furthermore, calculated measures relate only to the fact class where they were specified, whereas calculated measures and object predicates in semOLAP are independent of the fact class as long as the necessary structure is provided by any target fact class. This is possible since our approach is not directly based on the relational data model, unlike Polaris [11], instead it is based on the multidimensional data model. In comparison to the Polaris approach, we support reusing and editing instantiated semOLAP queries to match the new information demand. SemOLAP queries can, additionally, be used as the data input for other semOLAP queries since each result is representing a possible fact class as it consists of measures and dimension attributes.

### 7 SUMMARY AND FUTURE WORK

We have proposed a guided query composition approach based on semOLAP patterns. The semOLAP pattern approach provides the conceptual foundation to allow ad hoc query composition by domain experts. This conceptual foundation is realized by a data-centric and model-driven implementation which guides the domain expert during the instantiation of the FPEs. For each FPE instantiation, bindings are recommended by considering the semOLAP knowledge graph which comprises knowledge about the semantic schema, the pattern structure, and the current FPE binding. Therefore, basic relations are introduced which are used to check structural dependencies between the (subtypes of the FPE ranges. This allows users to move through the FPE instantiation process and select recommended bindings which consider the current instantiation state, the FPE’s type information, and constraints of the FPE itself.

For each semOLAP pattern a default navigation flow is calculated and provided to guide the user through the instantiation process. If an instantiation sequence other than suggested by the default navigation flow is more convenient for a particular pattern, a custom navigation flow can be configured by a developer. To this end, basic relations are extended to bidirectional relations, allowing to recommend bindings independent of the navigation sequence. Even navigation flows between FPEs which are not represented in the FPE’s dependsOn relationships can be supported, hence, leading to a maximum level of flexibility. The drawback of this flexibility is that instantiation situations can occur where no bindings at all can be recommended, since an unsupported FPE value combination is selected.

Future work will include displaying available information currently hidden in the semantic representation of schema elements, e.g., the measure SumOfMilkYieldParlour is linked to the AGROVOC 6 ontology which includes a concise definition of the measure. The calculation of a navigation flow can also consider the number of available FPE values. This would require the consideration of the selectivity of FPE values, i.e., it is preferable to start with the FPE value specification which has the highest potential to reduce the number of potential values of other FPEs. Furthermore, the visualization of the result, which is currently limited to a table representation, will be extended. In the future domain- and data-dependent visualizations will be automatically applied to the result. This visualization will also comprise a generated caption as well as a generated result description.

### REFERENCES


International Workshop on Big Data
Visual Exploration and Analytics

BigVis

Nikos Bikakis, ATHENA Research Center, Greece
George Papastefanatos, ATHENA Research Center, Greece
Olga Papaemmanouil, Brandeis University, USA
Enabling Hierarchical Exploration for Large-Scale Multidimensional Data with Abstract Parallel Coordinates

Gaëlle Richer, Joris Sansen, Frédéric Lalanne, David Auber, Romain Bourqui
{gael.le.richer,joris.sansen,frederic.lalanne,david.auber,romain.bourqui}@u-bordeaux.fr
LaBRI, Univ. Bordeaux, CNRS, UMR5800 France

ABSTRACT
As data collection grows more common in various domains, there is a call for adapted or newer methods of visualization to tackle magnitudes exceeding the number of available pixels on screens and challenging interactivity. Exploratory visualization of large data present two major challenges: perceptual scalability and processing scalability. The first is concerned with overcoming the fundamental limitation of screens and human perception. The second deals with efficiently processing large volumes of data to achieve responsive interactions. Multiscale visualizations are an effective technique for solving the first challenge that builds on several levels of data abstraction to provide the user with an initial overview and subsequent incremental detail. The focus of this paper is on multidimensional data, a ubiquitous form of data among large-scale data sets, and parallel coordinates, a representation largely used for this type of data. For this representation, defining abstractions and interactively generating levels is not straightforward. Building upon several previous aggregated parallel coordinates representations, we propose a unifying and thinking model for conceiving and describing multiscale parallel coordinates and their interactions. Using this formalism, we present a focus+context representation which bounds the number of visual items with a fixed resolution parameter while supporting exploration up to the item-level. Processing scalability is addressed by carrying out computation in a distributed manner on a remote data-intensive infrastructure. Bounding the visual items ensures perceptual scalability but also bounds the data transfer between this infrastructure and the rendering client.

KEYWORDS
visualization, large-scale data, parallel coordinates, multiscale visualization, distributed computing, focus+context

1 INTRODUCTION
Multidimensional data, a form of data common in many domains, encompasses all lists of individuals composed of several attributes (possibly temporal). Multidimensional items are studied from several aspects: the particular behavior of individual items relative to the whole, the relationship between values from two dimensions and the distribution of values along each dimension [7]. Parallel coordinates, introduced by Inselberg and Dimsdale [19], is a well-known technique of visualization for such data. Each item is represented by a polyline which anchors on axes are positioned at the corresponding attribute value of the item (see Fig 1b). Axes are traditionally aligned in parallel forming a sequence of two-dimensional subplots sharing one axis with their predecessor. Usual interactions are axis reordering to analyze relationships between all dimension pairs and selection to trace subsets of items across axes. Selection relates to interactive means of choosing subsets of items and enhancing them such that they can be discriminated from the rest.

In this paper, we are interested in supporting the interactive exploration of multidimensional datasets with a large number of items and a moderate number of dimensions. We built upon the assumption that large datasets, scaling to billions of records, lead to major overplot when displayed using the traditional line-based parallel coordinate representation. Indeed, as the number of records increases, the plot becomes cluttered and analyses may be hindered as specific patterns are concealed [10]. Heinrich and Weiskopf [18] listed several clutter reduction approaches for parallel coordinates. One solution lies in the display of aggregates computed per-dimension as in [20, 29, 32] instead of single items (see Fig 1c). Despite the aggregation, these abstract plots successfully provide overview information similar to a traditional plot and perceptually scale for any size of input data. Rendering time and transfer time in client-server architectures are also reduced since they are bounded by the number of aggregates. Sansen et al. [32] leveraged this bound by precomputing the result of several interactions and thus clear the dependency of linear scans of the data upon interaction. Indeed, past a certain number of records, data cannot fit in a desktop computer memory and linear scans over the data affect performance more negatively since they either involve reads on disk or network transfer between several computing units.

As for any abstract visualization, deeper analyses are limited by the amount of information conveyed by the visual aggregates. Two types of interactions can alleviate this limitation: changing the level of detail (show more details) and adjusting the aggregation (show different details). Supporting these interactions is essential but strongly increases the number of states of the visualization and reasonable precomputing and storing of all these states as performed by Sansen et al. [32] is challenging. The processing scalability challenge here has been addressed by Rübel et al. [31] on a modern high-performance computing (HPC) platform. HPC platforms are large and expensive computing systems suited for highly complex and real-time computation. They are composed of multiple processors connected through a fast network and use fast memory. As such, they are particularly
adapted to tightly coupled tasks where several processors work on the same task and exchange data. Distributed systems, on the contrary, are networks of computing units, usually commodity hardware, connected in a shared-nothing architecture (memory and storage are independent to each unit). On these systems, data transfer between computing units uses a slower network connection and thus is critical for performances. Consequently, they are most adapted to loosely coupled data-parallel tasks on large amounts of data. In addition to being cheaper alternatives for data-intensive computing, they offer easier horizontal scalability (allocation of additional computing units) as their hardware and architecture are less sophisticated. The filtration and aggregation problems at stake in abstract parallel coordinates are data-parallel tasks. We focus on these less expensive and more accessible infrastructures to address them.

Our aim is to enable interactive exploration down to the item-level over large-scale multidimensional data with an abstract parallel coordinate plot. The main contributions of this paper are summarized as follows: (i) a unifying model for abstract parallel coordinate plots based on hierarchical aggregation, (ii) formalization of multiscale and regular interactions in this model, (iii) a focus+context representation with bounded visual items and drill-down/roll-up interactions, extension of the work of [32], (iv) an example implementation using a distributed infrastructure with components that benefits from horizontal scalability.

The paper is organized as follows: we first present previous work on perceptual scalability in parallel coordinate plots and their interactions with a focus on multiscale approaches (Sec. 2). Then, we describe the proposed graph-based formalism for abstract parallel coordinates based on hierarchical aggregation (Sec. 3). In Sec. 4, we study how different interactions are expressed in this model. This yields a prototype implementation and design of a scalable parallel coordinate plot based on hierarchical aggregation, using a so-called big data infrastructure. This visualization is described and its scalability evaluated in Sec. 5. Finally, in Sec. 6 we draw conclusions and present directions for future work.

2 RELATED WORK

Recent works have focused on the scalability of visualization applications for large-scale data with different techniques, among which are: data reduction, multi-threading, GPU-acceleration, and incremental or approximate data processing. In the case of visualization, the scalability of a system often refers to its capacity to accommodate and handle growing amounts of data. Handling massive datasets brings about two main challenges for exploratory visualization: perceptual scalability and processing scalability as noted by [12, 16, 24]. The first is concerned with the legibility of visualizations representing numerous items relative to the space available on a screen (so-called screen real-estate problem) and human capabilities to apprehend them. The second relates to the computational cost of processing numerous items on each user input, that can create latencies responsible for decreasing user performances [23]. A taxonomy of different techniques regarding the perceptual scalability aspect was established by Ellis and Dix [10]. A general solution is data reduction, which can be categorized into two approaches: either representing a subset of the data items (sampling, filtering) or meta-items (aggregation, mathematical models). Several work proposed methods (called multiresolution, multiscale, hierarchical or even stratified) for navigating through multiple levels of detail supported by precomputation ([15, 24, 30]). This work is related to general techniques addressing perceptual and processing scalability for interactive parallel coordinates.

2.1 Perceptual Scalability in Parallel Coordinates

Various approaches have been proposed to improve the legibility of parallel coordinate plots by either reducing the clutter produced by the multiplicity of overlapping and crossing lines or enhancing their patterns. Approaches can be categorized into geometry-based relying on computer graphics techniques and data reduction approaches that use approximation or summary of the data. Geometry-based approaches display all items with shape or position modifications to alleviate the overdraw in-between axes, for instance by bundling lines (e.g. [33]). These techniques have the advantage of producing no or few loss of information but have the drawback of still being prone to over-plot since no reduction of the number of displayed items is performed. Data reduction approaches limit the number of visual items either by sampling items [9] or using meta-items. Model-based approaches mathematically reduce the data to a continuous function, and meta-items usually represents the density of the underlying data (e.g. [27]). Aggregation approaches have been presented for parallel coordinates through different schemes: aggregation of dimensions (e.g. [2]), aggregation of items or values (e.g. [13, 28, 29]) or combinations of both (e.g. [21, 26]).

In this paper, we are interested in scalability relative to the number of items, not dimensions, hence we focus on aggregation over items and their values. Previous work using aggregation have used kernel density estimation, independently applied to dimension [29], different hierarchical clustering algorithms over multidimensional items or dimension (e.g. [13]) but also binning on two-dimensional sub-spaces (e.g. [14, 28]). Aggregates have been represented by their statistical properties: extrema (e.g. [3]), cardinality (e.g. [20]), mean (e.g. [13]), and/or other metrics [29].

As reflected by previous work, multiple dimensions allow for different levels of grouping: the item level (dimensional), the value level (per-dimension), the line level (two-dimensional). For aggregation-based methods, the value level has the advantages of losing less information, preserving continuity on axes (potentially broken by two-dimensional aggregation) as well as an idea of the relationship between dimensions (broken by item-level aggregation).

2.2 Multiscale Parallel Coordinates

Abstract representations are usually conditioned by a parameter that controls the resolution of the display. Multiscale visualizations allow navigation between multiple levels of abstraction, using a drill-down/roll-up interaction (respectively for increasing and decreasing levels of detail). Elmqvist and Fekete [11] provided general guidelines for multiscale representations and interactions, based on such hierarchical aggregation. Bikakis et al. [4] presented a framework for hierarchical aggregation oriented towards the computational aspects of hierarchical navigation. Interactively changing the level of detail can be integrated in different ways (see [8] for a general review). A first approach, similar to geometry zooming, either (1) filters part of the representation to maintain a fixed number of visual items on screen or instead (2) purposely displays increasing number of items (e.g. [6, 29]). Filtering has the advantage of being scalable when the amount of displayed items is controlled; however, the overview and context
are lost along the way. One drawback of displaying an uncontrolled number of items is that the screen pixel limitation could always be reached for a sufficiently massive dataset and information would be lost. Additionally, from a certain point it may overwhelm human capacities. Consequently, zooming without filtering does not allow interactive exploration to the item-level complying with perceptual scalability. For zooming with filtering, an alternative to the loss of context is the overview+detail approach which displays both the filtered zoomed view and an overview (e.g. dimension zooming of [13]).

A second approach, called focus+context, consists in displaying heterogeneous levels of detail: a selected portion of the data is shown with greater details to the detriment of the rest. This approach allows to gain detail locally while preserving the overview. Fisheye lenses used in the work of [25] are an example applied in screen space. In data space, [13] and [28], although using different aggregation strategies, presented techniques where a subset of items can be enhanced and displayed with fine-scale details layered over the rest of the data, abstracted to some level.

To the best of our knowledge, none of these solutions propose to interactively change the level of detail locally, in a focus+context fashion, while bounding the number of visual items.

2.3 Processing Scalability of Multiscale Parallel Coordinates

For the past ten years, large-scale or massive has been used to qualify increasingly big data, now up to tera or petabytes in size [12]. For multidimensional data, above about 10^5 items, with a dozen of dimensions, perceptual scalability and processing scalability becomes problematic for interactive analysis. Abstract representation and precomputation of interactions are solutions respectively addressing perceptual and processing scalability, as [32] presented. In addition to the enhancement of subsets of items and the reordering of axes, abstract representations require a drilling interaction to allow fine data analysis.

Processing scalability can be tackled by parallelism, distributed processing and precomputing of complete or partial results for instance. For parallel coordinates, most interactions can be applied to two-dimensional subplots independently and on separate portions of the dataset without needing any communication (they are pleasantly parallel). This property has been exploited by Rübel et al. [31] on a HPC platform, and [32] on a distributed platform. Rübel et al. [31] addressed both scalability challenges with an histogram-based representation adapted from [28]. Histograms are two-dimensional aggregations of the data, precomputed in parallel, potentially for different resolutions. On more affordable hardware, Sansen et al. [32] addressed the same challenges with a parallel sets representation for an Hadoop+Elasticsearch ecosystem that also relies on full precomputation of certain types of interactions. Both works presented good scalability evaluations relative to the number of computing units used. However, these techniques only support the display of balanced levels of detail i.e. drill-down is global and necessarily increases the number of displayed items. With this approach, gaining detail at the item-level does not scale to large datasets.

Figure 2: Rationale behind the graph model. (a) Example data. (b) Opacity-based parallel coordinates. (c) Parallel coordinates matrix where all 2-dimensional subspaces are represented. (d) Many-to-many plot. (e) Graph-based representation G(X). Shadowed areas cover identical values from the same dimension and form the partition R. (f) Quotient graph G/R(X) (cardinality mapped to size).

3 A GRAPH-BASED FORMALISM FOR ABSTRACT PARALLEL COORDINATES

Consider $X \in \mathbb{R}^{n \times d}$, a set of multidimensional quantitative data with $n$ items and $d$ dimensions. On a traditional parallel coordinate plot, only a fraction of the input data subspaces is represented. Fig. 2c and 2d show two examples of representations that display all subspaces at once: respectively the parallel coordinate matrix [17] and the many-to-many plot [22]. The model we propose integrates all subspaces without assumption on the layout of axes and supports abstraction. A value in the matrix inherently belongs to a tuple (or item), and a dimension. We focus on abstraction based on aggregation of tuples and do not directly address aggregation of dimensions. The rationale behind this approach is that dimensions hold semantic meaning not embedded in their numerical content. Therefore, automatically and meaningfully aggregate them is not straightforward: some types of values may not make sense aggregated together. Consequently, we target datasets that challenge scalability by their size in tuples (more than millions of items) and hold moderate amounts of well-chosen dimensions.

3.1 Abstraction in Parallel Coordinates

In parallel coordinates, the values of the input matrix are connected to each other by lines to materialize the tuple or tuples they belong to. Extending this metaphor, an $n \times d$ matrix can be seen a graph linking its values based on the relation induced by tuples. This undirected graph is composed of $n$ separate cliques.
of $d$ vertices, that is, $n$ complete graphs (see Fig. 2e). We note this graph $G = (V, E, w)$, where $V$ denotes the vertices, $E$ the edges, $w$ the weighting function, $w : V \rightarrow \mathbb{R}$ that stores vertex values. We note $D = \{v_i \in [1, d]\}$ the partition of $V$ that groups vertices along their origin dimension. Given a dimension sequence, a parallel coordinate plot is built by taking the subgraph of $G$ composed of only the edges joining vertices of consecutive dimensions in the sequence. Vertices of this subgraph are positioned on their corresponding axis along a vertical position given by the weighting function. Shadowed regions on Fig. 2e represent a vertex partition $R$ that groups identical values of the same dimension. Such partition is a refinement of the partition $D$. Merging vertices of $G$ belonging to the same subset of a partition $R$ corresponds to taking the quotient graph of $G$ relative to $R$, noted $G_{/R}$. Two subsets $S, S'$ in the partition $R$ are represented by adjacent vertices in $G_{/R}$ if, and only if, some vertex in $S$ is adjacent to a vertex in $S'$ in $G$. Further, two aggregation functions are defined to compute the properties of the meta-nodes and meta-edges, i.e. vertices and edges of the quotient graph. Aggregation functions return a tuple of weights given a set of edges or vertices as input. Weights (cardinality, extremum, mean, etc) provided by such functions are subsequently used for assigning visual properties to meta-edges and meta-nodes. For instance, on Fig. 2b, the opacity of lines on the parallel coordinate plot depends on the cardinality of meta-edges. On this example, the aggregation of identical values allows to find the optimal number of lines to draw. Indeed, the aggregation done here in data-space relates to the one done in screen-space when rendering the polylines with alpha composing.

Given some aggregation functions, the same process is applied to obtain an abstract representation from any valid partition $R$ of $V$, that is, refinement of $D$. Fig. 3 presents an example of partition obtained with per-dimension clustering and another type of representation based on extremum values. Tuple-based approaches are incorporated into the model by refining the input tuple partition into a refinement of $D$. Each subset of the tuple partition can be decomposed into $d$ groups along their membership in $D$. Fig. 4 shows an example of tuple partition refinement and a tuple-oriented representation. Using our model and the introduced notations, the abstract representation is fully defined by: (i) a valid partition of the matrix values, (ii) two aggregate functions assigning weights to meta-nodes and meta-edges, (iii) an axis layout and visual encodings for meta-node and meta-edge weights (vertical position, color, etc).

3.2 Hierarchical Abstraction in Parallel Coordinates
Hierarchical aggregations are precomputation of different levels of abstraction that naturally support multiscale representations. Such precomputation outputs a rooted tree structure whose leaves are the data objects to aggregate. Every node of a rooted tree is said to cover the leaves of the subtree it is the root of. An antichain (also called tree cut) in a tree is a set of nodes $S$, such that no node of $S$ is an ancestor of another node of $S$. An antichain is maximal if it cannot be expanded by any other node without violating the antichain property. The subsets covered by the nodes of a maximal antichain form a partition of the tree leaves: the antichain property makes them pairwise disjoint, the maximal property ensures that their union is the set of leaves.

In our model, an abstraction is directed by a partition refining the partition $D$ which is equivalent to $d$ partitions, one for each $V_i$ of $D$. By augmenting each subset $V_i$ with a tree, such partition can be defined by one maximal antichain for each tree. This is equivalent to defining a single maximal and non-trivial antichain in the unified tree where the root’s children are the roots of all dimension trees. We note this unified tree $T(V)$, and call its direct subtrees dimension hierarchies.

Overall, for some input data modeled by the clique graph $G = (V, E, w)$ and its associated hierarchy $T(V)$, an abstraction is defined as previously described with the valid partition of $V$ induced by a maximal antichain in $T(V)$. This unifying approach accommodates multiple layouts of parallel coordinates (see Fig. 2) and several abstract parallel coordinates representations. We presented two abstract representations: per-dimension aggregation on Fig. 3d that corresponds to the technique presented by Palmas et al. [29] and on Fig. 4 tuple aggregation like presented by Fua et al. [13]. For tuple aggregation, the hierarchy defined on tuples is used to form all dimension hierarchies. Additionally, to represent the same tuple aggregates on all subplots, dimension hierarchies should all be cut at the same level.

4 COMMON INTERACTIONS
We take advantage of the hierarchical graph model described in the previous section to explore several hierarchical operations
and usual operations for parallel coordinates and investigate the incremental computation they require.

4.1 Level-of-Detail Operations

The drill-down and roll-up operations correspond to changes in the displayed level of detail. They respectively mean moving deeper in the hierarchical aggregation to show finer details, and upper to present a coarser display of the data. Since the model rests upon precomputed hierarchies of dimension values, the weights (cardinality, extrema, etc.) of all possible meta-nodes, i.e. nodes of the hierarchies, can be precomputed as well. The number of edges of a quotient graph with respect to a maximal antichain is always less than those of the original graph and generally greatly depends on the properties of the hierarchies (arity, depth).

In general, the number of maximal antichains is exponential. For example, complete binary trees, that are rooted binary trees with every level full except the last, have \( \Omega(2^n) \) maximal antichains where \( n \) is their number of leaves. This makes the precomputation of all possible quotient graphs, i.e. all meta-edge weights, with respect to every possible maximal antichain for given dimension hierarchies not possible for the scale of data we aim to tackle.

In traditional parallel coordinates plots, and in many derived techniques, each two-dimensional subplot is independent of the others and can be computed and displayed separately. When axes hold visual elements, like frequency plots, the corresponding visual items are usually mirrored and the two composing parts conceptually included in both sides subplots. If no edge is precomputed, the cost of an abstraction can be decomposed into independent substeps corresponding to all subplots, each requiring aggregation over \( n \) edges, one for each tuple.

In the context of hierarchical aggregation, gaining detail can be defined locally as the substitution of a visual aggregate for its children. In our model, the displayed level of detail is given by a maximal antichain of the hierarchy. Drilling-down on a meta-node replaces it in the current antichain by all of its children which still forms a maximal antichain of the tree. This entails computing the outgoing edges of these children since they are not precomputed. It corresponds to aggregating the tuples covered by the drilled meta-node. The opposite operation, rolling up or collapsing, is the replacement of sibling nodes with their parent in the antichain.

Globally increasing detail may result in an uncontrolled number of elements on display, independently of the arity and depth of hierarchies. We detail three drilling approaches that tackle this issue by trading context detail for scalability.

**Detail & Filter.** One approach is to define drilling on a meta-node as a filtering operation. This corresponds to computing the expansion of the meta-node over a filtered clique-graph, induced solely by vertices adjacent to the leaves it covers in the original clique-graph. This method effectively bounds the number of aggregates on an axis by the arity of the dimension hierarchy, i.e. the maximal number of siblings in the tree. This also bounds the number of edges in an abstract plot. The limit of this approach is that all subplots are modified since the weights of all meta-nodes and meta-edges from the quotient graph are to be updated to account for the filtering. Thus for each subplot, it costs a pass over as many bottom-level edges as the drilled meta-node covers items.

**Budgeted Detail.** A second approach lies in constraining the definition of maximal antichains such that their size comply with a predefined budget. In this setting, drilling potentially implies modifying the current maximal antichain in multiple points such that previously acquired detail collapses to allow drilling-down when the budget would have been exceeded otherwise. If these changes are restricted to a single dimension hierarchy, the incremental changes only involve one or two subplots. Computing these subplots costs a pass over \( n \) bottom-level edges for each, in addition to the cost of finding a suitable maximal antichain.

For a budget defined in number of visual items, the minimal budget allowing to define a maximal antichain that does contain a leaf node depends on the arity and depth of the tree. In a tree \( T \) with \( n \) leaves and arity \( a \), a maximal antichain containing at least one leaf and minimal in size can be expected to have between \( \text{depth}(T) \) and \( a \cdot \text{depth}(T) \) nodes. Considering the toy example of binary trees which nodes all have either 0 or 2 children, the minimum depth is \( \lceil \log_2(n) \rceil + 1 \) and the maximal depth is \( n \). Consequently, a visual budget allowing gaining detail up to the leaf level has to be chosen with respect to the depth of the hierarchy.

In our context, since this depth can reach orders of magnitude same as the number of tuples, this method does not scale without strong constraints on the properties of hierarchies.

**Dynamic Context Aggregation.** An alternative solution to collapsing nodes when focusing on deeper nodes is to aggregate them dynamically. The meta-nodes from the chosen antichain that are not in focus (i.e. not the deepest in the hierarchy) are represented aggregated which lowers their impact on the visual item budget. In general, these dynamically-computed aggregates, do not correspond to existing nodes in the precomputed hierarchy. This approach limits the visual items on an axis to the siblings of each focus and the aggregates of the rest of the meta-nodes that forms the context. Thus, for a dimension hierarchy of arity \( k \), an axis with \( f \) foci corresponds to at most \( k \cdot f + (f + 1) \) visual nodes, whatever the depth of these foci in the hierarchy is. Forming these aggregates induces an aggregation of their outgoing edges, therefore it similarly reduces the number of meta-edges. Consequently, with additional constraints on the arity of dimension hierarchies and the number of concurrent foci on an axis, this approach enables drill-down up to the deepest level while complying with a visual budget. Contrary to the previously described approaches, dynamic context aggregation matches the requirement of our system and thus is the one implemented (see Sec. 5.1 and 5.2).

4.2 Other Operations

**Axis Reordering** can be conceived either as the computation of a full plot or, incrementally, as an edition of one or more subplots which requires one or two passes over \( n \) bottom-level edges, for \( n \) the number of tuples. **Subset Selection** relates to the emphasis of a subset of tuples in a given representation. In an aggregate visualization setting, an abstraction can be computed over the subset of tuples and the resulting weights displayed over the complete abstraction using a discriminating encoding. Using this approach, a subset selection of \( m \) tuples requires going through \( m \) bottom-level edges per subplot, one per tuple, to compute their contribution to the current meta-edge weights. The process is the same for meta-node weights.
5 FOCUS+CONTEXT REPRESENTATION & SCALABLE IMPLEMENTATION

Our goal is to enable hierarchical exploration in abstract parallel coordinates while complying with the following scaling properties. On the representation-side, exploration should be possible down to the item level, in a top-down manner, while the number of visual items on display should be bounded for any size of input data. On the processing-side, network transfer latency between the displaying unit and the computing unit should be bounded and traditional operations (axis reordering, subset selection) should be supported. In the following, we present first the cornerstone of our method: the budgeted number of visual items. Then, we detail a novel focus+context display with intuitive drill-down capabilities using the dynamic context aggregation. And finally, we describe technical details and performance evaluation of the proposed system.

5.1 Bounded Number of Visual Items

Bounding visual items is essential for perceptual scalability but also ensures that data transfer between our rendering client and back-end unit remains bounded in size thus predictable in time. To this end, the properties of dimension hierarchies are constrained and the number of foci restricted to one per dimension.

Hierarchy Constraints. We use a user-defined \( k \) value that acts as a resolution parameter, bounding the number of visual items per displayed axis. This parameter is enforced as the maximal arity of dimension hierarchies. \( k \) should be chosen large enough for the representation to preserve an appropriate amount of information but small enough for the visual items to fit the available screen space. Additionally, dimension hierarchies should order their leaves with respect to their values such that every maximal antichain defines partitions of dimension intervals since the chosen visual representation relies on this property. Binning (equal range partitioning) and adaptive binning (equal size partitioning) are examples of partitioning algorithms that can be applied in a bottom-up fashion to produce hierarchies complying with these two requirements. No depth constraint is required, we adopt the dynamic context aggregation strategy to bound the number of vertices from quotient graphs displayed.

Level-of-Detail Navigation. We define as focus nodes, the nodes that have the maximal depth in the current antichain. The rest of the antichain nodes from one dimension are aggregated into the minimal number of context nodes such that their order is preserved. In the proposed implementation, the top-level nodes are initially presented as focus nodes and each drill-down triggers a dynamic aggregation (examples are presented in Fig. 5). Consequently, there is no more than \( k \) focus nodes and two context nodes at once per axis. Thus, the number of nodes on display is bounded by \( k + 2 \) per axis and the number of edges by \( (k + 2)^2 \) per subplot.

5.2 Visual Encoding

We propose an extension of the work presented in [32] which included two visual encodings: one oriented towards the distribution of tuples, the other towards the distribution of values themselves. Basically, the first encoding maps aggregate height to their cardinality, while the second maps it to their covered interval size (distance between extrema). We extend the encoding and positioning of aggregates to emphasize the focus regions, composed on each dimension of the data displayed with the finest detail. Context regions are displayed with slightly lower width to emphasize the focus regions. Since regions of interest are incrementally refined by successive drill-down, they get smaller in height relative to the whole both in terms of cardinality and interval of values. Therefore, the height of focus and context regions are rendered with different scaling factors. Focus nodes are represented colored and augmented by a bidirectional smoothed histogram. Context nodes are augmented by a pile of level blocks (see Fig. 5c). The wider these blocks are, the lower in the hierarchy are the nodes they represent (notice on Fig. 5c the difference in width of the outer blocks from the top and bottom context). The height and vertical order of level blocks follows the same encoding as those of focus nodes. Level blocks are computed on the client side and thus not transferred nor counted in the visual item budget.

In the drilling mode, focus nodes and level blocks are clickable. Clicking on a focus node triggers its animated expansion which split it into its children and merges its siblings into level blocks. Level blocks represent an aggregation of nodes previously in focus, thus they enable going back to this precise previous state.

5.3 System Overview

Our system consists of two main parts and follows a client/server architecture where the client is the visualization endpoint and the server is an interface to an on-demand computing and preprocessing back-end. The latter could be implemented both in a distributed environment for a Hadoop cluster and as a multi-threaded application for single machines (desktop computer or dedicated server). Past a certain number of input records (for a
given number of dimensions and resolution parameter), a distributed platform should be more efficient while facilitating load expansion.

The client is a WebGL application that displays the representation, computes level blocks and context aggregation on node drill-down, and queries the supporting back-end for other interactions. The back-end server is a long-lived Spark application which runs distributed job on demand while keeping prepared data in memory. It computes dimension hierarchies in a pre-computation step and stores the resulting meta-node weights (extrema and cardinality) for all the hierarchies in a distributed database. Contrary to the previous system, the hierarchical aspect makes the number of displayable meta-edges too large to allow their precomputation in reasonable time (see Sec. 4.1).

The membership of each input data value is stored in a hierarchy matrix of the same size as the input data, where each value holds the list of computed ancestors for the matching input data value. This matrix is kept in memory and split among computing units which will pass over their slice of the data to filter and aggregate results on demand. The aggregation outputs the meta-edges (source, target and cardinality) as well as the weights of context nodes since they do not exist in the precomputed hierarchies. Upon user interaction and if necessary, the client requests the server which in turn runs a distributed operation and merges the partial results returned by computing units. Finally, the client receives the incremental changes in plain text and updates the view consequently.

5.4 Performance Evaluation

Our system supports moving an axis, selecting an aggregate node or edge, drill-down on a focus node and rolling-up by clicking on a context part. Drill-down, roll-up and moving an axis correspond to the same low-level operation which consists in computing the edges of two-dimensional subplots given the current focus on each dimension involved. Compared to the other operations that only modify two to three subplots, selection operations affect all subplots, thus are more computationally expensive.

Since we are interested in supporting $n \gg d^k$ datasets, we evaluate our system for varying number of tuples $n$ and using the most expensive interactive operation: node selection of the largest node (in covered tuples). To demonstrate the scalability of the approach we also evaluate performance relative to the resources allocated for computation.

Each node of the distributed platform has 64GB RAM and 2x6 hyper-threaded cores at 2.1GHz each, connected via a 1Gbit/s network. The single computer used for running the multi-threaded implementation has 2x4 hyper-threaded cores at 3.3GHz and 64GB RAM. Test datasets are generated for varying $n$ (from $10^6$ to $10^{10}$), with $d = 15$ and $k = 31$. Test datasets are generated such that pairs of dimensions present a close to null correlation factor [5] which tends to create close to the maximum number of edges between dimensions. Dimension hierarchies are generated using Canopy clustering [1] applied in a bottom-up manner.

On Fig. 6a, execution times of the selection of the largest node are plotted for increasingly large datasets. We evaluate both a multi-threaded single-computer implementation and the distributed Spark implementation. The stairs-shaped curve of the single computer can be explained by the similar trend of the selected node cardinality for every test dataset. Indeed, due to the hierarchy constraints and the bottom-up approach for clustering, the number of nodes on the top-level varies and the cardinality of the largest nodes does not increase linearly with $n$. We observe that, on our infrastructure, under $n = 2 \cdot 10^9$ (about $10^8$ selected tuples) the Spark application performances are stable despite increasing workload. This suggests that execution time for datasets of smaller size is dominated by costs related to network and disk I/O, and/or by merging all executor results by the driver unit. Indeed, the cost of merging results remains approximately the same as it is a function of the output size and the number of tasks running in parallel. Another visible aspect of the results on small datasets is their high variation. The oversized allocated memory can be an explanation: when triggered, garbage collection incurs a substantial delay. Overall, for $n$ less than approximately $10^8$, the distributed infrastructure underperforms the single-computer despite having more resources. Past this limit, the system is better tuned and seems to scale linearly relative to the number of tuples in the selection. This observation holds for the other experiments carried out for edge selection, roll-up, and drill-down.

We investigate the scalability of the system using a dataset larger than this limit. Fig. 6b presents the median execution time for node selection on a $2 \cdot 10^9$-tuple dataset with varying numbers of executors. The plotted ideal execution time corresponds to a linear speedup, that is, halving the execution time when doubling the number of executors. This experiment shows that the system performs close to the ideal.

6 CONCLUSION

We presented a graph model for hierarchical aggregation and interaction strategies for parallel-coordinate-based visualization. This model formalizes aggregation over multidimensional data at
the most-expressive level by making use of per-dimension hier-
archies. This approach treats all dimensions equivalently which
matches the way dimensions are handled in parallel coordinates.

Based on this model, we presented a client/server system for
interactive exploration of large multidimensional data using ab-
stract parallel coordinates. We address the limitation of the pre-
vious system, based on fixed partitions of dimension values, with
hierarchies that allow to interactively change partition. The pro-
posed drill-down operation enables the definition of an arbitrary-
detailed focus region on each axis while retaining context in a
reduced form. On the client-side, we showed how to display
focus and context regions with intuitive navigation cues. The
strength of this approach is that it supports exploration down to
the item-level while controlling the number of visual items, thus
ensuring perceptual scalability.

On the server-side, the back-end processing is handled on a
distributed platform with components that scale horizontally. Be-
sides drill-down/roll-up operations, the implementation supports
standard parallel coordinate operations. Experimental results
demonstrate the scalability of the back-end system relative to the
size of the input data and to the resources allocated for computa-
tion. The results indicate that the proposed system can support
increasingly large datasets by expanding its network of comput-
ing units. To a certain extent, adding computing resources can
also reduce interaction latencies.

One focus of the design is the bounded data transfer between
the client and server parts which relies on the single-focus ap-
proach (on each axis) and the precomputation of dimension hi-
erarchies with a bounded arity k of small orders of magnitude.
The hierarchies being predefined allows for efficient filtering and
aggregation of items based on ancestors but may be limiting for
exploration. As a counterbalancing measure, future work could
add support for user-driven hierarchy refinement. Depending on
the scope and scale of the refinements this could be handled on
the client-side and/or enforced on the server-side.

Another path for future work is methods for latency reduction
other than horizontal scaling. Space is a possible trade-off. Even
if the number of possible meta-edges makes their total precom-
putation infeasible, partial precomputation could be investigated:
either beforehand or as a background process targeting meta-
edges that are the most likely to be requested given the current
state.

ACKNOWLEDGMENTS
This work has been carried out as part of the 'REQUEST' (PIAO18062-
645401) and "SpeedData" (PIAO17298-398711) projects supported
by the French "Investissement d'Avenir" Program (Big Data –
Cloud Computing topic). The authors would also like to thank
the anonymous referees for their valuable comments and helpful
suggestions.

REFERENCES

high-dimensional data sets with application to reference matching. In Knowledge
Discovery and Data Mining. ACM, 169–178.

integrating hierarchical dimensions into parallel coordinates visualisations. In Proc. of Int. Conf. on Knowledge Technologies and Data-driven Business. ACM.


[16] P. Godfrey., J. Gryz, and P. Lasek. 2016. Interactive Visualization of Large Data

EuroVis/Short Papers (2012).


Finding the Needle in a Haystack: Entropy Guided Exploration of Very Large Graph Cubes

Dritan Bleco
Athens University of Economics and Business
Athens, Greece
dritanbleco@uueb.gr

Yannis Kotidis
Athens University of Economics and Business
Athens, Greece
kotidis@uueb.gr

ABSTRACT
Graphs provide an elegant and versatile solution for modeling complex datasets, especially when the focus of the analysis is on highlighting interesting associations between data entities. Graph cubes permit analysis of the resulting data graphs at various levels of granularity based on their node and edge attributes. In this work, we utilize information entropy measures in order to help the analyst navigate within the rich information contained in a graph cube. Our metrics suggest navigations (drill-downs) towards more detailed data descriptions, conditioned on what has been observed at a coarser resolution. We propose a graph analysis workflow that first suggests interesting cuboids from the exponential collection of aggregations that exist in the graph cube. At a latter step, this workflow handpicks sub-graphs out of these aggregations that deviate significantly from the rest of the data. We experimentally validate our techniques using real datasets and demonstrate that the proposed entropy-based exploration can help eliminate large portions of the respective graph cubes from consideration. Our techniques help locate the “needle in the haystack” and steer the user towards data skew hidden within vast valleys of near-uniform interactions.

1 INTRODUCTION
Despite their versatility, graph data have specific characteristics that make their analysis often challenging. Of particular interest in graph data are the relationships between nodes captured by the edges of the graph. These relationships should be analyzed with respect to attribute values available at the nodes and edges. For example, a data scientist may want to investigate how users of a social network, depending on their gender, relate to other users based on their nationality. This inquiry can be accommodated by aggregating existing relationships (edges) in the data graph based on the attributes of their constituent nodes. This process forms a graph cuboid, as is depicted in Figure 1.

The graph cube contains all such possible cuboids that can be generated given the raw graph data [6, 10, 15, 22, 35]. As in the case of the data cube [11, 12, 16, 28], there is an exponential number of aggregations that define the space of all possible such cuboids. Moreover, each of these cuboids is not a flat relation, but a complex property graph filled with intrinsic structural information based on the formed relationships and annotated with computed summary statistics over the attributes of the graph nodes and edges. A data explorer, familiar with the simpler multidimensional framework of data cubes, may be overwhelmed when she tries to navigate this data deluge.

In this work, we model the relationships between the graph cuboids as a graph cube lattice produced by taking the Cartesian product of simpler data cubes on the attributes of the nodes and edges of the data graph. Using this model, we propose a graph cube analysis workflow that can be used to explore interesting associations hidden within very large graph cubes. Our suggested workflow utilizes two intuitive entropy measures, introduced in [5], in order to reveal associations that deviate from the expected behavior. The first measure termed as external entropy permits us to suggest certain drill-down navigations that reveal associations that deviate from what has already been observed at the higher-level aggregations of the graph cube. As demonstrated by our experiments, from the exponential possible navigations in the graph cube, only a very small percentage of them leads to interesting observations. The external entropy helps the data explorer navigates towards interesting cuboids in the graph cube lattice and may be used to prune a significant portion of the lattice from consideration.

In a second step of the workflow, we utilize entropy calculations in order to elevate particular data associations that deviate from the rest of the relationships within the cuboids selected from the first step. This is achieved by using an internal entropy metric that helps the analyst elevate aggregate interactions that are the result of skew in the data graph. These interactions become prominent when the raw data is aggregated at the levels denoted by the cuboid under investigation.

In our experimental section we present results of utilizing our techniques while processing real social datasets of realistic sizes. We compare our techniques against an alternative method that prunes parts of the graph cube based on a minimum support threshold, as in association rule mining. We observe that our framework maintains the most varied parts of the data distribution independently of their frequencies. Thus, many interesting trends revealed by our technique that focuses on data skew within and across cuboids, would be missed by methods that merely seek frequent patterns. We also discuss prominent trends revealed by our techniques on the real datasets used.

2 MOTIVATING EXAMPLE
We consider a social network which depicts relationships between different users. Each user can be represented as a node in a graph. Each user profile has three attributes: gender (male, female), nation (Greece, Spain, France) and profession (doctor, professor, musician). For brevity, we refer to these attributes values by their initial letter. Each edge in the data graph is associated with a numeric value that indicates the number of interactions between the respective users.

A possible inquiry on this network is to examine how users depending on their gender, relate to other users based on their nationality. To accommodate this query we need to perform three different aggregations. First, starting nodes (i.e. nodes with outgoing edges) are grouped into two aggregate nodes corresponding...
to gender values male and female, respectively. Similarly, three aggregate nodes corresponding to nations Greece, Spain and France are formed. Finally, each edge of the network, depending on the gender attribute value of its starting node and the nation attribute value of its ending node, is aggregated into an edge between the corresponding aggregate nodes created at the previous steps. At this time, a desired aggregate function can be computed. In this example, we assume that this function is SUM(). The resulting aggregate graph is depicted in the middle of Figure 1. Based on its construction we refer to it as the (gender - nation) cuboid.

Continuing with the running example, the cuboid on the left part of the figure depicts the outcome of drilling-down from (gender - nation) to the (gender, profession - nation) cuboid. The intuition is that we would like to explore whether the profession of the source node, in addition to its gender, affects the number of observed relationships. In this contrived example, the aggregated edges from cuboid (gender - nation) are split almost evenly when drilling down to the (gender, profession - nation) cuboid. Thus, this particular navigation step does not seem to reveal interesting correlations for this data, conditioned on what is already observed in the (gender - nation) cuboid.

On the right part of Figure 1, we depict another possible drill-down, this time to the (gender, nation - nation) cuboid. In this new context, some interesting irregularities are revealed. First, while female users are linked evenly to users from Greece and Spain, when these links are conditioned based on her nationality we can see that females from Spain are mainly linked to users from the same country. Similarly, French males are mostly linked to users from Spain. Thus, while cuboid (gender - nation) suggest a uniform relationship based on the nationality of the target node, cuboid (gender, nation - nation) reveals that this is not true for certain members of the user community. It is worth noting that the majority of the links in the (gender, nation - nation) cuboid still follow the same uniform pattern suggested by the (gender - nation) cuboid, since most links emanate from female users in Greece and male users in Spain. Thus, the examples discussed above are exceptions to what is suggested by the (gender - nation) cuboid. These are depicted in red color inside the (gender, nation - nation) cuboid.

3 THE GRAPH CUBE

In our running example, each user profile has three attributes, namely gender (G), nation (N) and profession (P). If we treat these attributes as dimensions in OLAP analysis, the resulting data cube has $2^3 = 8$ possible cuboids. The work of [35] extended the data cube framework to work on graph data by considering also the relationships between aggregated graph nodes. In particular, consider a data cube for the data attributes of the starting nodes in the graph and another one for the ending nodes. These data cubes share the same dimensions and are, thus, identical in structure (i.e. contain the same set of cuboids). The graph cube can be considered as the Cartesian product of these two data cubes: of the starting- and the ending-cube. In this running example, a graph cuboid can be $((gender, nation), (\cdot, nation), \cdot)$ or, for brevity, $(gender, nation - nation)$. The starting nodes on this cuboid are aggregated graph nodes based on their gender, nation attribute values. Similarly, the ending nodes are aggregations of raw graph nodes based on the nation attribute values. Starting and ending nodes in this cuboid are interconnected according to the raw graph edges. These raw data edges are consolidated producing a graph cube edge along with a measure. The user may choose any combination of functions based on attributes on the constituent nodes and edges.

In many applications, edges of the data graph may have attributes that can also be treated during exploratory analysis as dimensions. Attributes on the edges of the data graph can be aggregated creating yet another set of cuboids in an edge-cube lattice. For example, in a social network a connection can have several attributes like the type $T$ of the relationship (family, friend, sibling etc.) and the date $D$ that this connection was established. Naturally the analyst may want to include those attributes and observe their interaction with the node attributes. As an example, let us consider the case where the data graph edges have a Type ($T$) and a Date ($D$) dimension (the latter being rolled-up in a suitable level, e.g. day, year or month). The edge-cube lattice in this example contains four cuboids, namely $(*, T, D)$, $(T, D)$. These cuboids can also participate in the Cartesian product of the graph cube computation adding another dimension in the final cube. A cuboid in this extended cube is denoted as $(\text{starting node-aggregation} \cdot \text{edge-aggregation} \cdot \text{ending node-aggregation})$.
4 USING ENTROPY TO NAVIGATE THE GRAPH CUBE

4.1 Main concepts

In this work, we present techniques that help the analyst identify irregularities when navigating different aggregations of the original data graph. Because of the exponential number of cuboids in the graph cube, it is extremely difficult to manually explore all possible cuboids and all navigation steps among them (roll-up, drill-down) in search for interesting patterns. This realization provides the motivation for our framework. We seek to provide the analyst with solid mathematical tools derived from information theory and in particular the information entropy, that will help her reveal interesting irregularities.

In [5] we introduced two types of entropy calculations. The first one measures the significance of a whole cuboid and it is called external entropy. This type of entropy is used to detect whether a drill-down process during exploratory analysis to a more detailed cuboid provides additional insights or not. In our running example, external entropy calculations on the (gender, profession - nation) and (gender - nation) cuboid will suggest that no apparent irregularities are revealed by this drill-down and it can, thus, be omitted. In contrast, the external entropy metric will suggest that the drill down to the (gender, nation - profession) cuboid reveals certain skew in the calculated relationships that deviate from what is expected by observing the relationships in the (gender - nation) cuboid. The second type is the internal entropy that evaluates the relationships inside a cuboid. Internal entropy can help steer the user towards surprising, skewed relationships (such as those depicted in red in the figure) within a large cuboid, eliminating relationships that do not reveal trends that deviate from the expected behavior.

In what follows, we first introduce the suggested entropy calculations used in our navigation framework. More details on these metrics can be found in [5]. We discuss a graph cube analysis workflow that can be used for processing very large graph cubes.

4.2 External Entropy Metric

The edges from a cuboid $C_i$ can be represented as a virtual relation. Each record in this virtual relation is associated with (i) a set of attribute values $s_1, \ldots, s_j$ derived from the starting nodes of the corresponding edge, (ii) a set of values $e_1, \ldots, e_w$ derived from the ending nodes and (iii) an aggregate value $a$ that denotes the result of the selected aggregate function applied over the selected measures from these constituent nodes and edges. In the example of Figure 1, edge (female, Spain) of cuboid (gender - nation) will be mapped to a single row (female, Spain, 310) in the virtual table. Each such record $r_j=(s_1, \ldots, s_j, e_1, \ldots, e_w, a)$ can be viewed as a discrete probability distribution $P(s_1, \ldots, s_j, e_1, \ldots, e_w)$ by normalizing the aggregate $a$ value on each record by the sum of all aggregate values in the instance of the relation. Thus, record $r_j$ is associated with a probability value $p(a_j) = \frac{a_j}{\sum_{i=1}^{m} (a_i)}$. In our example, the probability value for the record that maps to edge (female, Spain) will be $\frac{310}{104+310+77+11=720}$. The external entropy ($eH$) of a cuboid is defined as the negative of the logarithm of the probability distribution of the records in the virtual relation ($m$ in the formula bellow refers to the number of edges in the cuboid that also equals the number of records in the virtual table).

$$eH(C_i) = -\sum_{j=1}^{m} p(a_j) \log_2 p(a_j)$$ (1)

A drill-down process in the graph cube lattice is triggered by adding another attribute (starting or ending) in cuboid $C_i$. This leads the analyst to another more detailed cuboid $C_k$ at the next level of the lattice. We refer to cuboid $C_k$ as the “child” of $C_i$, while $C_i$ is the “parent” of $C_k$. While drilling down from the parent $C_i$ to the child $C_k$ we can calculate the delta-entropy, i.e. the difference between the two external entropies as:

$$\delta_{i,k}(C_i, C_k) = eH(C_k) - eH(C_i)$$ (2)

The delta entropy is a non-negative number. This is because the external entropy of the child cuboid $C_k$ is greater or equal to the external entropy of its parent $C_i$. The maximum external entropy of the child is obtained when the aggregate $a$ of each edge is distributed evenly among the more detailed edges in $C_k$ and their number is maximized. Let $d_{max}$ denote the number of possible values of the attribute on which the drill down process was performed. In order to maximize the entropy of a child cuboid, an edge with aggregate value $a^j$ in $C_i$ is replaced during the drill-down with $d_{max}$ more detailed edges in $C_k$ with aggregate values $a^j_{0} = \frac{a^j}{d_{max}}$. Thus, the maximum possible external entropy value

Figure 2: The graph cube when both node attributes (side data cubes) and edge attributes (middle data cube) are being used. The graph cube lattice is produced by taking the Cartesian product of the three data cube lattices that form the constituent data cubes.
of the child cuboid given its parent is
\[
eH^\text{max}(C_k) = -\sum_{j=1}^{m} p(a^j_i) \cdot \log_2 \frac{p(a^j_i)}{d_{\text{max}}} \tag{3}
\]

The external entropy rate quantifies how informative, the process of drilling down from parent \(C_i\) to its child \(C_k\) is:
\[
eH_{\text{rate}}(C_k, C_i) = \frac{eH(C_k) - eH(C_i)}{eH^\text{max}(C_k) - eH(C_i)} \tag{4}
\]

This rate takes values between 0 and 1. A value that is close to 1 implies that the drill-down process doesn’t change significantly the distribution of the records and, thus, no new insights are given to the analyst. The exact opposite happens when the value is close to, or zero. We can therefore exclude less interesting navigations in the lattice by defining a maximum external entropy rate threshold value between zero and one. When the external entropy rate of a drill down navigation step surpasses the threshold, then this drill down is omitted from consideration.

4.3 Internal Entropy Metric
With similar arguments we can introduce an internal entropy rate threshold in order to select subgraphs within a cuboid that differ significantly from the rest of the cuboid data. Since we consider directed data graphs, we distinguish between two kinds of internal entropy, namely starting internal entropy and ending internal entropy.

Consider cuboid \(C_i\) with \(l\) distinct combinations of starting attribute values of the form \((s^y_1, s^y_2, \ldots, s^y_l)\). Let \(m_y\) be the sum of the aggregate values of all such edges, where \(y \in \{1, l\}\). For each such combination (indicated by parameter \(y\)) there are \(f_y\) edges with different combinations of ending attribute values.

Let \(z_{yk}\) be sum of their aggregate values as well. We calculate the starting internal entropy as the conditional entropy of the ending attributes’ values conditioned from each starting attribute combination of values.
\[
siH(c^y_i) = -\sum_{j=1}^{f_y} p(q^y_j) \cdot \log_2 \frac{p(q^y_j)}{m_y} \quad \text{where} \quad p(q^y_j) = \frac{z_{yk}}{m_y} \tag{5}
\]

The ending internal entropy \(eH\) is defined in an analogous manner. As in the case of external entropy, we introduce the internal entropy rate (for the starting or ending internal entropy, respectively) as the fraction between the (starting/ending) internal entropy and the maximum possible value of internal entropy.

The value of the internal entropy rate is between 0 and 1 and can be used to select the most prominent trends within a cuboid, as will be explained in the next Section.

5 GRAPH CUBE ANALYSIS WORKFLOW

Motivated by the examples of the previous subsections, in this work we present techniques that

- Weigh all possible navigations within a graph cube lattice and suggest drill-down operations that reveal surprising trends, conditioned on what is observed in the more abstract cuboids contained in the cube. This process eliminates a significant portion of the graph cube, steering the user towards cuboids that reveal skew that is hidden when focusing in more abstract aggregations.
- Evaluate the relationships within the cuboids suggested from the previous step in order to reveal parts of data that contain skewed relationships.

In Figure 3 we depict the distinct steps involved in using our techniques for analyzing massive graph data cubes. After the graph cube is computed, we first utilize an external entropy rate threshold in order to prune edges of the lattice and, consequently, cuboids that do not provide significant insights with respect to their ancestors and descendants. For those cuboids that are connected by edges suggested by this process, we compute the internal entropy rates (for starting and ending attributes aggregated at the level denoted by the corresponding cuboid). We can then use a user-provided internal entropy rate threshold to only return relationships in these cuboids that do not exceed the threshold or, we can sort them and return the top-\(k\) selections in increasing order of internal entropy rate.

6 EXPERIMENTS

In this section, we provide preliminary results from applying our suggested framework on three real social network datasets. The focus on this exposition is to first highlight the pruning power of using entropy to navigate very large graph cubes and then to discuss some of the main trends observed in the social datasets used.

The datasets used are summarized in Table 1. The Twitter dataset was crawled by our team and contains 3 attributes: gender, location and language, used in each user profile. We also crawled the VK dataset from VKontakte, the largest European online social networking service. The sample contains 5 attributes: birthyear, country, city, gender and education level of the user. Finally, the Pokec dataset, available from [20] is a social-network from Slovakia and uses 6 node attributes: age, region, gender, registration year, public profile and completion percentage of the profile.

In order to compute the graph cubes of these datasets, we set up a small cluster of 4 PCs equipped with Intel i7-3770 CPUs clocked at 3.40GHz, 4GB of memory and 1TB 7200rpm HDDs. We used the popular Apache Spark [34] framework on 8 VMs (one being the master) running on this cluster. The graph cube for each dataset was computed using an extension of the BUC algorithm discussed in [5].

In first experiment, we utilize the suggested data analysis workflow and evaluate the pruning power of the external and the
internal entropy metrics. Figures 4a, 4b and 4c illustrate how the starting internal and external rates reduce the number of records of the graph cube, in each dataset. The plots for the using the ending internal entropy are similar and are omitted due to lack of space. The plots suggest a steep reduction in the sizes of the graph cubes for all datasets, as the respective entropy rate thresholds are increased. We observe that using thresholds in the ranges from 5% to 20% helps trim the million or billions (in the case of the Pokec dataset) records in the corresponding graph cubes to manageable sizes. This suggests that indeed, in these real data, there is a needle in the haystack that begs to be revealed. This is more evident in the largest graph cube from the Pokec dataset that contains 4096 cuboids and more than 66 billion records. In that dataset, a 10% external entropy threshold leads the analyst to focus on less than 0.002% of the aggregated graph cube records that contain 9 out of the 10 more prominent associations (when ranked in decreasing order of their internal entropy).

In Figures 5a, 5b and 5c we depict the filtered sub-lattices (sets of cuboids) selected when using an external rate threshold of 3.5% in the graph cube analysis workflow of Figure 3. For the Twitter dataset 17 out of the 64 cuboids of the graph cube are chosen. For the VK dataset 9 out of 1024 cuboids are retained. Finally, for the Pokec dataset only 10 from the 4096 cuboids are kept for post-processing. Based on the characteristics of the datasets shown in Table 1 we observe that the external entropy helps prune more cuboids when the number of node attributes is increased, as this results in larger lattices for the full graph cube.

These filtered cuboids are used as input for the final stage of our workflow that further selects parts of these cuboids based in their internal entropy. For that step we used a rate threshold of 20% and present in Table 2 some characteristic results for each
dataset. Due to space limitations the attributes in the table are shown with their first letter. Thus, N stands for nation, L for language, G for gender, A for age and E for education level.

In the Twitter dataset, we find that users from all countries follow mostly users from the USA. Exceptions include users from Portugal, Romania, Latvia, Venezuela, Taiwan, Chile, Brunei, Brazil and Norway. Users of these countries seek to follow mainly other users from the same country. From the cuboid (nation - gender) the entropy reveals that users from Monaco and Nauru follow males 2.2 times more often than females. Similarly, users from Thailand follow men 1.7 times more often than women. On the contrary, Mongolia users follow women 2.1 times more often than men.

From the VK dataset, we mine some other trends. Most connections are towards 35-year-old users from Russia and after that from Ukraine. Most connected users are born between 1986 and 1990. Users from USA are connected mostly with women, the same appears for users from Kazakhstan. Users connected with Turkish profiles are 70% men. Women are related uniformly with both genders while men are connected 60% with other men and 40% with women. Most users are connected to other profiles without a university degree and after that with users that got their diploma between 2008-2012.

Using the entropy-based techniques in the Pokec dataset we see other interesting trends. First, we observe that most relationships are towards women. Specifically, users between 19 and 22 years old have mainly connections to women. On the other hand, 19-year-old females are more frequently connected with other females. With respect to location, connections between the same cities dominate. Also, the most connections are with users from the Presovsky kraj and Presov regions. Users from most of the regions are connected with female users except for those from Nitriansky kraj and Nitra that are associated with more men. 19-year-old users from Presovsky kraj, Bardejov are connected mainly with male peers. Users between 32 and 37 years old from Banskoobystricky kraj, Banska are connected mainly with females that are 22 years old.

The rightmost column of Table 2 depicts the support of the corresponding trend. The numbers validate our intuition that skewed trends are quite often hidden within valleys of uniform behavior. Indeed, most trends have small support values and would be, thus, missed by a frequent itemset counting algorithm.

7 RELATED WORK
The work in [35] introduced the graph cube that takes into account both attribute aggregation and structure summarization of the underlying graphs. This work is mainly focused on cuboids
that aggregate the starting and ending nodes on the same dimensions, e.g. (nation - nation). More general aggregations that differentiate between the starting and ending nodes of the graph are not specifically mentioned but can be addressed under a cross-cuboid computation that is mentioned as an extension. In our work, we elevate such cuboids as first-class-citizens in the graph cube framework. As our experiments with real datasets indicate, such cuboids often hold significant insights for the underlying interconnections. Another distinction is that the work of [35] considers all records in the proposed graph cube. As we show in our work, only a small part of a complex graph cube carries interesting information when analyzed under the lens of our entropy-based navigation framework.

A recent work [33] considers aggregate attributed graphs. The authors name their model as a hyper graph cube and show how to compute it using MapReduce batches. The hyper graph cubes aggregate separately attributes at vertices and edges and then calculate the Cartesian product between them. Thus, they do not exploit and analyze the existing relationships under different levels of aggregation on the starting and ending nodes of the graph. OLAP-style summarization in the context of RDF graphs has been recently studied in [2]. The most significant difference from the previous works in graph cubes, is that our techniques address the vast size and complexity of the produced cuboids. To the best of our knowledge we are the first that utilize the entropy in order to filter the information of a graph cube.

The authors of [24] propose a novel framework for reconstructing multidimensional data from stored aggregates using the maximum entropy principle. In a nutshell, the proposed technique finds the model with the least information (maximum entropy) given a set of constraints that can be the $2^n - 2$ different aggregations in the cube (excluding the raw data and the grand total aggregate). The method uses a multi-pass algorithm called Iterative Proportional Filtering (IPF) that converges to the maximum entropy solution. The information entropy was first introduced in [29] as a measure of unpredictability of information content. It measures how much information there is in an event. Entropy is frequently used for splitting decisions when computing Decision Trees [27] The information gain measures the change in information entropy from a prior state to new state after a split. Our external entropy rate measure utilizes the information gain metric in the nominator of its respective formula but differs in that it also takes into consideration the maximum possible increase in the entropy of a child cuboid in a drill down step. By conditioning the information gain over this quantity we are able to obtain the bounds that our selection algorithm utilizes.

Recently, an entropy-based model has been proposed [25] in order to estimate the strength of social connections by analyzing users’ occurrences in space and time. This work considers triplets of (user, location, time) data and utilizes entropy to measure the diversity of user co-occurrences. In our work, we utilize

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Trend</th>
<th>Cuboid</th>
<th>$\min(siH_{rate}, eiH_{rate})$</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Twitter</td>
<td>* - En</td>
<td>N - L</td>
<td>11.05%</td>
<td>87.12%</td>
</tr>
<tr>
<td>Twitter</td>
<td>* - USA</td>
<td>N - N</td>
<td>12.15%</td>
<td>36.07%</td>
</tr>
<tr>
<td>Twitter</td>
<td>Portugal - Portugal</td>
<td>N - N</td>
<td>15.01%</td>
<td>0.025%</td>
</tr>
<tr>
<td>Twitter</td>
<td>Latvia - Latvia</td>
<td>N - N</td>
<td>15.23%</td>
<td>0.006%</td>
</tr>
<tr>
<td>Twitter</td>
<td>Venezuela - Venezuela</td>
<td>N - N</td>
<td>15.73%</td>
<td>0.009%</td>
</tr>
<tr>
<td>Twitter</td>
<td>Taiwan - Taiwan</td>
<td>N - N</td>
<td>16.10%</td>
<td>0.006%</td>
</tr>
<tr>
<td>Twitter</td>
<td>Chile - Chile</td>
<td>N - N</td>
<td>16.39%</td>
<td>0.029%</td>
</tr>
<tr>
<td>Twitter</td>
<td>Brunei - Brunei</td>
<td>N - N</td>
<td>16.88%</td>
<td>0.001%</td>
</tr>
<tr>
<td>Twitter</td>
<td>Brazil - Brazil</td>
<td>N - N</td>
<td>16.89%</td>
<td>0.564%</td>
</tr>
<tr>
<td>Twitter</td>
<td>Norway - Norway</td>
<td>N - N</td>
<td>17.03%</td>
<td>0.061%</td>
</tr>
<tr>
<td>Twitter</td>
<td>Monaco - Male</td>
<td>N - G</td>
<td>17.31%</td>
<td>0.002%</td>
</tr>
<tr>
<td>Twitter</td>
<td>Nauru - Male</td>
<td>N - G</td>
<td>17.71%</td>
<td>0.00004%</td>
</tr>
<tr>
<td>Twitter</td>
<td>Thailand - Male</td>
<td>N - G</td>
<td>17.93%</td>
<td>0.021%</td>
</tr>
<tr>
<td>Twitter</td>
<td>Mongolia - Female</td>
<td>N - G</td>
<td>18.06%</td>
<td>0.001%</td>
</tr>
<tr>
<td>VK</td>
<td>* - 35, Russia(Ukraine)</td>
<td>* - A, N</td>
<td>13.23%</td>
<td>0.963%</td>
</tr>
<tr>
<td>VK</td>
<td>* - [1986..1990]</td>
<td>* - A</td>
<td>14.11%</td>
<td>2.388%</td>
</tr>
<tr>
<td>VK</td>
<td>Female - USA</td>
<td>Kazakhstan</td>
<td>G - N</td>
<td>15.53%</td>
</tr>
<tr>
<td>VK</td>
<td>Male - Turkey</td>
<td>G - N</td>
<td>16.42%</td>
<td>0.082%</td>
</tr>
<tr>
<td>VK</td>
<td>Male - Male</td>
<td>G - G</td>
<td>16.51%</td>
<td>37.74%</td>
</tr>
<tr>
<td>VK</td>
<td>* - No Diploma</td>
<td>Diploma 2008-2012</td>
<td>* - E</td>
<td>17.01%</td>
</tr>
<tr>
<td>Pokec</td>
<td>age[19..22] - Female</td>
<td>A - G</td>
<td>12.97%</td>
<td>0.698%</td>
</tr>
<tr>
<td>Pokec</td>
<td>age[19, Female - Female</td>
<td>A - G</td>
<td>13.05%</td>
<td>0.001%</td>
</tr>
<tr>
<td>Pokec</td>
<td>Cityx - Cityy (same city connection)</td>
<td>L - L</td>
<td>15.66%</td>
<td>9.908%</td>
</tr>
<tr>
<td>Pokec</td>
<td>Female - Male, Male - Female</td>
<td>G - G</td>
<td>15.98%</td>
<td>65.34%</td>
</tr>
<tr>
<td>Pokec</td>
<td>* - Presovsky kraj</td>
<td>Presov region</td>
<td>* - R</td>
<td>16.18%</td>
</tr>
<tr>
<td>Pokec</td>
<td>Nitrianskykraj, Nitra - Man</td>
<td>R - G</td>
<td>16.47%</td>
<td>0.001%</td>
</tr>
<tr>
<td>Pokec</td>
<td>age[19, Presovsky kraj</td>
<td>Bardejov - Male</td>
<td>A, R - G</td>
<td>17.02%</td>
</tr>
<tr>
<td>Pokec</td>
<td>age[32..37], Branska - Female, 22</td>
<td>A, R - G, A</td>
<td>17.11%</td>
<td>0.012%</td>
</tr>
</tbody>
</table>

Table 2: Main trends derived from the three social datasets
entropy to measure the diversity within and across graph cuboids. The works of [3, 4] consider the case of analyzing very large collections of smaller data graphs, while in this work we consider a single massive graph that is under investigation.

Our techniques can be used in conjunction with existing systems for parallel graph processing [30] and tools like Peruces [19] that summarizes an input graph using statistics such as PageRank, radius, degree and flags outlier nodes [31], graph visualization tools [18], or with systems that recommend promising visualizations on aggregated datasets like SEEDB [32]. Our techniques may also be combined with the work of [13] that seeks intuitive drill-down operations from aggregated views of data.

Application of graph mining techniques [1, 8, 17, 21, 23, 26] is also orthogonal to our framework and can be used in conjunction. For instance, the work of [23] looks for structural patterns (or motifs) in the k-hop neighborhood of a node. The work of [21] suggests aggregation of graph nodes scores on vertices that contain some attribute of interest. Unlike conventional iceberg queries, the authors propose an aggregation method that is based on random walks and demonstrate their effectiveness and scalability. The authors of [7] explore data mining techniques to analyze tagging behavior on social graphs. The authors of [9] introduce graph-pattern association rules (GPAR). These rules extend traditional association rules with graph patterns that specify association between entities in a social graph.

There is recent work on systems that permit interactive exploration of very large data cubes. For example DICE [14] is a distributed system that utilizes faceted exploration in order to limit the number of possible queries in an interactive session. Extending this technique for graph cubes is an interesting research direction. Our entropy-based cube navigation framework can be combined with the idea of faceted exploration, either as a pre-processing step that limits the set of possible aggregations (cuboids) that need to be considered, or during interactive exploration by using the external/internal entropy rates in order to steer the user towards skewed correlations.

8 CONCLUSIONS

Graph data is becoming popular due to emerging applications that need to process and analyze interconnected datasets. In this work we proposed a graph data analysis framework based on the graph cube operator. Similar to the data cube, graph cubes contain an exponential number of aggregations of the raw data graph. Moreover, these aggregations are not simple flat records but rather complex graph structures that make their exploration cumbersome.

To overcome these obstacles our framework utilizes two novel entropy metrics that help locate unusual patterns hidden within billions of graph data aggregations. We put our framework to the test using three real social datasets of realistic size. Our preliminary results demonstrate that indeed entropy-guided exploration can help prune lots of uniform correlations enabling the analyst to focus on skewed parts of the data that often reveal interesting trends.

REFERENCES


[22] Xiaolei Li, Jiawei Han, and Hector Gonzalez. 2004. High-Dimensional OLAP: A Minimal Cubing Approach. In (eds)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004. 526–539.


[33] Zhengkui Wang, Qi Fan, Huiju Wang, Kian-Lee Tan, Divyakant Agrawal, and Anne El Abbadi. 2014. Pagrol: Parallel graph olap over large-scale attributed graphs in ICDE.


[35] Peixiang Zhao, Xiaolai Li, Dong Xin, and Jiawei Han. 2011. Graph Cube: On Warehousing and OLAP Multidimensional Networks. In Proceedings of ACM SIGMOD.
Map-Based Visual Exploration of Geolocated Time Series

Georgios Chatzigeorgakidis
University of Peloponnese
Greece
chgeorgakidis@uop.gr

Dimitrios Skoutas
Athena R.C.
Greece
dskoutas@imis.athena-innovation.gr

Kostas Patroumpas
Athena R.C.
Greece
kpatro@imis.athena-innovation.gr
g

Spiros Athanasiou
Athena R.C.
Greece
spathan@imis.athena-innovation.gr

Spiros Skiadopoulos
University of Peloponnese
Greece
spiros@uop.gr

ABSTRACT
The amount and significance of time series that are associated with specific locations, such as visitor check-ins at various places, have been claimed in many domains over the years. Although several works exist for time series visualization and visual analytics in general, there is a lack of efficient techniques for geolocated time series in particular. In this work, we present an approach that relies on a hybrid spatial-time series index to allow for interactive map-based visualization and summarization of geolocated time series data. In particular, we use the BTSR-tree index, which extends the R-tree by maintaining suitable bounds for the time series indexed at each node. We describe the structure of this index and show how it can be effectively exploited to produce map-based visualizations of geolocated time series at different zoom levels efficiently. We empirically validate our approach using two real-world datasets, as well as a synthetic one that is used to test the scalability of our method.

KEYWORDS
time series visualization, geolocated time series, visual exploration

1 INTRODUCTION
Time series are generated and stored at a vastly increasing rate in many industrial and research applications, including the Web and the Internet of Things, public utilities, finance, astronomy, biology, and many more. A significant portion concerns geolocated time series, i.e., those generated at, or otherwise associated with specific locations. While indexing, mining, and exploring time series data has attracted a lot of interest from the database and data mining communities [4, 12, 25, 28], studying of geolocated time series is still largely overlooked.

Geolocated time series can be found in various domains and applications. A typical example is encountered in the DAIAD project1, where time series are used to represent water consumption measured by smart meters installed in urban households. Analyzing such time series can provide valuable insights regarding trends and patterns of consumer behavior in daily life. Such results can then be used to forecast and balance water demand, as well as to plan and prioritize interventions that can guide consumers towards more sensible water use. Similar use cases can also be found in other domains, such as in geomarketing or mobile advertisement, where geolocated time series may represent the number of visitors or the revenue generated at a certain location across time. Extracting insights, trends and patterns can be significantly facilitated by map-based visualizations of summarized time series data. For example, such visualizations can reveal which type of consumption patterns are most frequently observed among consumers in a certain area or what the spatial distribution of sales for a certain product looks like.

However, time series is an inherently complex data type, and such datasets can reach extremely large volumes, both horizontally (i.e., very long series across time) and vertically (i.e., time series generated by countless sources). Consequently, management, analysis and exploration of such Big Time Series Data is a task of excessive complexity, requiring efficient algorithms. In particular, visual exploration of geolocated time series needs to process the required information in real time, while the user interacts with the map. Whenever the user zooms in or scrolls the map, the visual analytics and aggregates should be computed on-the-fly, e.g., identifying the predominant patterns in the time series and their spatial distribution within the map area.

Consider the example illustrated in Figure 1. When the user zooms the map into the red rectangle, the visualization tool should readily identify and present the two patterns (shown in blue and green color) appearing therein. To avoid cluttering the map, when the spatial distribution is very dense or the map area is too large, it is meaningful to display only aggregate information, e.g., the number of time series per identified pattern and their respective centroids; individual time series may be identified only at a greater zoom level.

Such fast computation and retrieval for large datasets of geolocated time series can be enabled by indexing. Several approaches have been proposed that efficiently index large amounts of plain time series data, either relying on Discrete Wavelet Transform like [5] to reduce dimensionality of time series, or the family of indices based on Symbolic Aggregate Approximation (SAX) over the time series [3, 4, 28, 31]. However, all aforementioned techniques index the data solely on the time series domain, not taking the spatial dimension into account. If the analyzed time series are inherently associated with a spatial attribute (e.g., locations of smart meters), such indexing does not efficiently support queries and visualizations that do not simply apply to the time series domain, but also involve spatial filters. As in the example of Figure 1, a user may need to explore time series similar to a specific pattern, but also having locations within the actual map area. For such mixed requests, it is inefficient to evaluate each predicate separately, e.g., using first the time series index to

1http://daiad.eu/
In summary, our main contributions are as follows:

• We propose an adapted variant of the BTSR-tree index as well as a novel algorithm for its traversal in order to quickly retrieve summaries (bundles) of geolocated time series within a given area.

• Thanks to robust index support, we introduce a method that can cluster bundles according to time series similarity, calculate their support and identify their spatial extent, thus enabling their interactive map-based exploration.

• We exemplify the proposed visualization method with two use cases based on real-world datasets.

• We also empirically evaluate the performance of index traversal, confirming its low execution cost against a large synthetic dataset of geolocated time series.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 outlines basic concepts and formulates the problem. Section 4 describes the BTSR-tree index and introduces our approach on summarization of geolocated time series for their efficient visual exploration. Section 5 presents indicative use cases with map visualizations and also reports performance results. Finally, Section 6 concludes the paper and outlines future research directions.

2 RELATED WORK

As our approach suggests visual exploration of time series and is based on indexing of such data, next we briefly survey both of these research topics.

Visual Exploration of Time Series. In constrast to declarative visualization specifications suggested in [29], a recent tutorial [21] advocates the use of example-based methods in exploration of large relational, textual, and graph datasets. Such a query-by-example approach has been applied in [13] so as to explore relevance feedback for retrieval from time series databases. Instead of returning the top matching time series, this technique incorporates diversity into the results, which are presented to the user for feedback and refined in several rounds.

RINSE [32] is a Recursive Interactive Series Explorer specifically designed for exploration of data series. Built on top of ADS+ [31], a special adaptive index structure for data series, it can progressively build parts of the index on demand at query time, concerning only those chunks of the data involved in users’ queries. In terms of visualization, users can get those series qualifying to range or nearest-neighbor queries interactively drawn on screen, as well as monitor various statistics regarding the index footprint (e.g., RAM and disk usage) as it gets updated.

In contrast, ATLAS [6] is a visual analytics tool specifically geared towards interactivity when ad hoc filters, arbitrary aggregations, and trend exploration are applied against massive time series data. This client-server architecture employs a column store as its backend equipped with indexing, and preemptively caches data that may be required in queries so as to reduce latency when panning, scrolling, and zooming over time series.

Recently, the ONEX paradigm [22] concerns online exploration of time series. It first constructs compact similarity groups over time series for specific lengths based on Euclidean distance, and then can efficiently support exploration of these groups with the Dynamic Time-Warping (DTW) method over their representatives of different lengths and alignments.

Besides, smoothing can be applied to streaming time series to remove noise in visualizations while preserving large-scale deviations [27]. To highlight important phenomena without harming representation quality from oversmoothing, this approach introduces quantitative metrics involving variance of first differences and kurtosis to automatically calibrate smoothing parameters.

ForeCache [2] leverages two prefetching mechanisms to facilitate exploration of large geospatial, multidimensional and time
series data stored in a DBMS. By predicting the user’s behavior, it fetches the necessary data as the user interacts with the application.

None of the aforementioned methods and systems provides map-based visual exploration of geolocated time series, as is the goal of our work in this paper.

**Indexing of Time Series.** Earlier approaches towards indexing of time series data were based on leveraging multi-resolution representations. For instance, the Discrete Wavelet Transform [15] is used in [5] to gradually reduce the dimensionality of time series data via the Haar wavelet [17] and generate an index using the coefficients of the transformed sequences. In [26], it is further observed that, other than orthonormal wavelets, bi-orthonormal ones can also be used for efficient similarity search over wavelet-indexed time series data, demonstrating several such wavelets that outperform the Haar wavelet in terms of precision and performance. In addition, an alternative approach regarding k-nearest neighbor search over time series data is introduced in [18]. The proposed method accesses the coefficients of Haar-wavelet-transformed time series through a sequential scan over step-wise increasing resolutions.

State-of-the-art approaches for time series indexing comprise methods based on the Symbolic Aggregate Approximation (SAX) representation [20]. This is derived from the Piecewise Aggregate Approximation (PAA) representation of a time series [19, 30], by quantizing the segments of its PAA representation on the y-axis. The first attempt to leverage the potential of the SAX representation was presented in [28], introducing the indexable Symbolic Aggregate Approximation (iSAX), capable of a multi-resolution representation for time series. The iSAX index was further extended to iSAX 2.0 [3] by enabling bulk loading of time series data. Its next version is the iSAX2+ index [4], which handles better the expensive I/O operations caused by the aggressive node splitting while building the index. Finally, the ADS+ index [31] is another extension of iSAX, which overcomes the still significantly expensive index build time by adaptively building the index while processing the workload of queries issued by the user. A comprehensive overview of the time series indexing approaches based on the SAX representation is presented in [23].

Unfortunately, none of the abovementioned access methods can inherently support geolocated time series, i.e., time series inextricably associated with a location. To the best of our knowledge, the only index in the literature that supports such time series is the BTSR-tree index [7]. This hybrid index follows a similar rationale set by *spatio-temporal indices* [8, 9, 11, 14] that have been proposed to speed up evaluation of queries combining location-based predicates with keyword search. Essentially, this paradigm implies combining a spatial index structure (e.g., R-tree, Quadtree, Space-Filling Curve) with a textual index (e.g., inverted file, signature file). Depending on their structure, these variants can be characterized either as spatial-first or textual-first indices [10]. In a similar spirit, our BTSR-tree is a spatial-first index based on the R-tree that can additionally abstract similarity of time series instead of a textual one. As a result, it can offer analogous improvements when searching against geolocated time series data, as we discuss in more detail in Section 4.1.

**3 PROBLEM DEFINITION**

Next, we introduce notation and definitions used in our approach, and we formally define the problem addressed in this paper.

A time series is a time-ordered sequence of values \( T = [v_1, \ldots, v_w] \), where \( v_i \) is the value at the \( i \)-th time point and \( w \) is the length of the series. In particular, we deal with time series that are additionally characterized by a location, denoted by \( T.loc \). Assuming a 2-dimensional space, we further use the notation \( T.loc_x, T.loc_y \) to refer to the \((x, y)\) coordinates of \( T \)'s location.

In the spatial domain, the distance between two geolocated time series \( T \) and \( T' \) of equal length \( w \) is calculated using the Euclidean distance of their respective locations. Furthermore, we normalize this distance with \( \text{maxDist}_{tp} \), i.e., the maximum spatial distance of any pair of objects in the dataset, to obtain a measure in the interval \([0, 1]\). Thus:

\[
\text{dist}_{tp}(T, T') = \frac{\sqrt{(T.loc_x - T'.loc_x)^2 + (T.loc_y - T'.loc_y)^2}}{\text{maxDist}_{tp}}
\]

In the time series domain, similarly to other prior works (e.g., [28]), we also apply the Euclidean distance to measure the similarity of a pair of objects. In future work, we plan to make use of more complex distance measures [24]. More specifically, we calculate the distance between two time series \( T \) and \( T' \) as follows:

\[
\text{dist}_{ts}(T, T') = \frac{\sum_{i=1}^{w} (v_i - v'_i)^2}{\text{maxDist}_{ts}}
\]

where \( \text{maxDist}_{ts} \) denotes the maximum distance of any pair of objects in the dataset and is used for normalization, as above.

To index and summarize time series, we use the notion of Minimum Bounding Time Series (MBTS). An MBTS is a summarization of a set of time series \( T \), defined by a pair of upper and lower time series bounds that contain them. Formally, given a set of time series \( T \), its MBTS consists of an upper bounding time series \( T_{up} \) and a lower bounding time series \( T_{lo} \), constructed by selecting the maximum (for \( T_{up} \)) and minimum (for \( T_{lo} \)) of values at each time point among all time series of the set as follows:

\[
T_{up} = (\max_{T \in T} T[0], \ldots, \max_{T \in T} T[w - 1])
\]

\[
T_{lo} = (\min_{T \in T} T[0], \ldots, \min_{T \in T} T[w - 1])
\]

We can now formally introduce our problem. Given a set of geolocated time series and an area, our goal is to produce a summary for visualization comprising the following two parts:

- **Time series summary:** A collection of MBTSs (bundles), summarizing the time series located within the given area.
- **Spatial summary:** A set of MBRs, each one associated with an object counter for each identified bundle.

The bundles provide a summarization of the time series that are contained within their MBTSs. Figure 2 depicts an example of two time series bundles for two different sets of time series. Regarding the spatial summary, for each MBRS associated with a certain bundle, the counter denotes the number of time series contained in it.

**4 APPROACH**

We propose a visualization method for geolocated time series that draws on a map the time series and spatial summaries for the current visible area. Using this process, a user can select the bundle of her preference and the proper spatial summary will appear on the map after acquiring the necessary MBRs from
the BTSR-tree index. Whenever the user zooms in/out or moves around the map, the BTSR-tree is traversed, and the corresponding bundles, MBRs and object counts are obtained to drive the visualization. In each case, the rectangle corresponding to the visible part of the map is used to feed a traversal algorithm that efficiently gathers the results. In the following, we describe this process in detail, after providing some necessary background information on the BTSR-tree index.

4.1 The BTSR-tree Index

To efficiently generate real-time visualizations of geolocated time series data, we need early access to both spatial and time series related information while traversing the index, in order to maintain low latency levels when drawing the required graphic elements. However, none of the approaches presented in Section 2 supports geolocated time series indexing. To the best of our knowledge, the recently proposed BTSR-tree index [7] is the only one that provides the desired functionality.

The BTSR-tree is based on the R-tree [16] for the spatial indexing part. The R-tree organizes a hierarchy of nested d-dimensional rectangles. Each node corresponds to a disk page and represents the MBR of its children or, for leaf nodes, the MBR of its contained geometries. The number of entries per node (excluding the root) is between a lower bound \( m \) and a maximum capacity \( M \). Query execution in R-trees starts from the root. MBRs in any visited node are tested for intersection against the search region. Qualifying entries are recursively visited until the leaf level or until no further overlaps are found. Several paths may be probed, as multiple sibling entries could overlap with the search region. The BTSR-tree extends the information stored within each node of the R-Tree with bundles of MBTSs. This allows to efficiently prune the search space when evaluating hybrid queries combining time series similarity with spatial proximity.

As in the standard R-tree, each node of the BTSR-tree has at least \( m \) and at most \( M \) entries and stores the MBRs of its children. Additionally, for each child, a node stores a pre-specified number of time series bundles, each consisting of an MBTS that encloses all the time series indexed in its subtree. Each bundle is calculated using Equation 3. Construction and maintenance of the BTSR-tree follow the procedures of the R-tree for data insertion, deletion and node splitting. Objects (i.e., geolocated time series) are inserted into leaf nodes, and any resulting changes are propagated upwards. Once the nodes have been populated, the bundles of each node are calculated bottom-up. To construct the time series bundles within each node, we rely on \( k \)-means clustering. The objects contained in each node are clustered according to their Euclidean distance on the time series domain. The example in Figure 2 depicts the bundles (the two bands with a thick outline) obtained for a set of time series (shown as thin polylines) when the number of clusters is set to \( k = 2 \).

For inner nodes, the bundles are constructed bottom-up. First, in each leaf node, the contained time series are clustered into \( k \) bundles. Then, the MBTS of each bundle is computed and stored in the node. As a next step, each parent node receives all the MBTSs of its children and computes its own \( k \) bundles and respective set of MBTS by clustering them. The process continues upwards, until reaching the root of the tree. Optionally, Piecewise Aggregate Approximation [19, 30] can be applied over the time series. As detailed in [7], this allows a trade off between the number of bundles per node and the MBTS resolution, thus permitting a larger number (> \( k \)) of bundles in nodes at higher levels in the tree hierarchy.

In addition, to support the required functionality of our visualization method, we further extend here the information stored in each node with the \textit{count} of geolocated time series that are fully contained within each bundle. This is also done bottom-up, while the index is traversed to calculate the bundles. At each leaf node, after the clustering, we propagate the number of members of each cluster to its parent, which, in turn calculates its clusters and aggregates the counts it has received for each bundle’s members. This procedure continues up to the root of the tree.

4.2 Summary Construction for Map-Based Visualization

We now present our summarization approach for producing map-based visualizations of geolocated time series. The process is outlined in Algorithm 1. It takes as input the \textit{query rectangle} \( q \), i.e., the area of the map for which the visualization is produced, and the number of bundles \( k \) to be generated. The process comprises three distinct steps. Initially, the BTSR-tree index is traversed to obtain the MBRs contained in the query rectangle, along with the bundles and the number of objects per bundle (Line 1). Next, \( k \)-means clustering is applied using the average time series per bundle as centroids (Line 2). Finally, the new bundles are calculated and the proper MBRs and corresponding object counts are assigned to each bundle (Line 3). Next, we describe each step in more detail.

\textbf{Step 1: BTSR-tree Traversal}. During this step, the BTSR-tree index is traversed, with the target being the fast provision of a predefined number \( k \) of geolocated time series bundles contained within the given area \( q \), along with the MBRs where these bundles can be found and the total number of geolocated time series that reside within each MBR. All required information is stored within the nodes of the BTSR-tree, thus, when a node that is contained within the query rectangle is found, the relevant information is retrieved and added to the intermediate results, without any need to continue searching in its sub-tree. The output of this step is passed to the next step of the procedure.

In more detail, the traversal is performed as follows. After initializing a queue with the root’s children (Line 7), we loop over it (Line 8) until it’s empty. For each inner node’s child \( N^i \), we check whether its MBR is contained within the given query rectangle \( q \) (Lines 11–12). If so, its MBR, time series bundles and
the number of objects per bundle are added to the intermediate results (Line 13) as a tuple with the following components:

\[
\langle \text{mbr}, \langle \text{mbts}_1, \text{cnt}_1 \rangle, \ldots, \langle \text{mbts}_k, \text{cnt}_k \rangle \rangle
\]

Each such tuple indicates the MBR of a node (mbr), consisting of the coordinates of the lower left and upper right point, as well as k pairs denoting the bundles of the node along with the corresponding number of objects per bundle. If the MBR is not contained in the query rectangle, we check whether it overlaps with it and if so, we add the child node to the queue (Line 15). If not, this MBR is located outside the query rectangle, and thus we can skip searching this child. Once no more nodes are left to search, the intermediate results are finally returned (Line 16).

**Step 2: Bundles Clustering.** The traversal algorithm returns tuples, each containing the bundles residing in the query rectangle, the corresponding nodes’ MBRs and the number of objects per bundle. During step 2, k-means clustering is executed on the average time series of each bundle.

Line 2 of Algorithm 1 calls the clustering procedure. Initially, for each tuple (Line 20), we loop over its bundles (Line 21) and generate a new tuple per bundle of the following format:

\[
(\text{T}_{avg}, \text{mbts}, \text{cnt}, \text{mbr})
\]

This new tuple contains an average time series, the bundle itself (mbts), the number cnt of objects enclosed in this bundle, and the MBR (mbr) this bundle belongs to (Line 23). The average time series \(T_{avg}\) is calculated by averaging the upper and lower bound of each bundle (Line 22), i.e., average value at each time point. The resulting collection of tuples (Line 24) is fed to the clustering procedure for generating k resultant bundles, corresponding MBRs and total number of objects per bundle. The final result (Line 35). This tuple has the following components:

\[
\langle \text{mbts’}, \langle \text{mbr}, \text{cnt} \rangle, \ldots, \langle \text{mbr}, \text{cnt} \rangle \rangle
\]

where mbts’ is a resulting bundle, along with the MBRs associated with it. Note that the number n of MBRs (as well as their shape) may be varying per bundle, reflecting the spatial distribution of the respective pattern. Each MBR is accompanied with the corresponding number of objects (i.e., raw time series) therein. The final result with all such tuples is then returned in order to generate the visualization (Line 36).

**Algorithm 1: Summarization of Geolocated Time Series**

**Input:** The query rectangle \(q\), the number of bundles to be generated \(k\)

**Output:** A list of tuples (Line 24) is fed to the clustering procedure. Initially, for each tuple (Line 20), we loop over its bundles (Line 21) and generate a new tuple per bundle of the following format:

\[
(\text{T}_{avg}, \text{mbts}, \text{cnt}, \text{mbr})
\]

This new tuple contains an average time series, the bundle itself (mbts), the number cnt of objects enclosed in this bundle, and the MBR (mbr) this bundle belongs to (Line 23). The average time series \(T_{avg}\) is calculated by averaging the upper and lower bound of each bundle (Line 22), i.e., average value at each time point. The resulting collection of tuples (Line 24) is fed to the clustering procedure for generating k resultant bundles, corresponding MBRs and total number of objects per bundle. The final result (Line 35).

5 EXPERIMENTAL EVALUATION

In this section, we first describe our experimental setup, followed by indicative examples of map-based visualizations of real-world geolocated time series, as well as scalability results using a synthetic dataset containing 4 million time series. The experiments were conducted on a Dell PowerEdge M910 with 4 Intel Xeon E7-4830 CPUs, each containing 8 cores clocked at 2.13GHz, 256 GB RAM and a total storage space of 900 GB. Finally, we assume that the index fits in memory.
5.1 Experimental Setup

5.1.1 Datasets. We use two real-world datasets selected from different application domains and with diverse characteristics. In addition, we generated a synthetic dataset to test the scalability of our method. Table 1 lists a summary of the characteristics of each dataset.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Area (km²)</th>
<th>Number of time series</th>
<th>Length $w$ of each time series</th>
</tr>
</thead>
<tbody>
<tr>
<td>Water</td>
<td>114</td>
<td>822</td>
<td>168</td>
</tr>
<tr>
<td>Taxi</td>
<td>2,500</td>
<td>4,179,960</td>
<td>168</td>
</tr>
<tr>
<td>Synthetic</td>
<td>114</td>
<td>4,000,000</td>
<td>168</td>
</tr>
</tbody>
</table>

DALAD Water Consumption (Water). Courtesy of the DAIAD project, we acquired a geolocated time series dataset of hourly water consumption for 822 households in Alicante, Spain from 1/1/2015 to 20/1/2017. In order to get a more representative dataset for our tests, we calculated the average weekly time series per household, which is the average consumption value per hour of the week. Thus, the length of each resulting time series is $24 \times 7 = 168$ values across the week.

NYC taxi dropoffs (Taxi). This dataset contains time series extracted from yellow taxi rides in New York City during 2015. The original data² provide pick-up and drop-off locations, as well as corresponding timestamps for each ride. For each month, we generated time series by applying a uniform spatial grid over the entire city (cell side was 200 meters) and counting all drop-offs therein for each day of the week at the time granularity of one hour. Thus, we obtained the number of drop-offs for $24 \times 7$ time intervals in every cell, which essentially captures the weekly fluctuation of taxi destinations there. The centroid of each cell is used as the geolocation of the corresponding time series.

Synthetic Water Consumption (Synthetic). To examine the scalability of our method, we generated a synthetic dataset comprising 4 million geolocated time series by inflating the water consumption dataset. This was achieved by using the original time series as seeds and introducing some random variations in their location and pattern. We chose the water dataset so as to generate a more densely populated dataset (Alicante is a medium-sized city), in order to stress-test our visualization method.

5.1.2 Parameters. We built the BTSR-Tree index setting the minimum and maximum number of entries per node to $m = 60$ and $M = 200$, respectively. Regarding the number of bundles, we set $k_0 = 5$ for its leaf nodes. The number of bundles for the traversal algorithm is set to be equal to the number of bundles at the leafs, i.e., $k = k_0 = 5$. For an evaluation of the BTSR-Tree index under different parameter settings, please refer to [7].

5.2 Map Visualizations

Our visualization method depicts the MBTS derived for the most representative patterns of time series at the currently visible area of the map. Once our summarization method returns the results, the corresponding MBRs contained in the current view and zoom level are drawn on the map, along with the number of the geolocated time series that belong to the selected bundle. This number is depicted using circles, colored green for small numbers, yellow for larger and red for more densely populated MBRs, thus easily conveying the local intensity of this pattern. The bundles are listed on the left of the map, using confidence bands to indicate their upper and lower bounds. The average time series of each bundle is also depicted. A user can scroll this list and select the bundle of their preference. Once a bundle is selected, the contents of the map are updated accordingly with the respective MBRs and aggregates.

Figure 3 shows an example of such visualization using the water dataset. The depicted area is in the center of Alicante, in the most densely populated zone of the city. In this example, Bundle 4 is selected (indicated with a green colored frame) and the relevant MBRs are shown on the map (using red colored frames). This indicates that inside each depicted MBR there exists a specific number of geolocated time series that have been clustered to the chosen bundle. As mentioned, each geolocated time series in this dataset represents hourly water consumption of a household across one week. Different consumption behaviors have been grouped together and a daily pattern for each bundle can be noticed which is due to the Circadian rhythmic way that people consume water [1]. The rather large number of geolocated time series in the bundle, considering the zoom level and the extent of the MBRs, intuitively suggests that neighboring families tend to have similar water consumption behavior.

Figure 4 illustrates another example, this time using the taxi dataset in New York City. This dataset is significantly larger, and the zoom level selected in this example is lower (a larger geographic area is visible), hence the MBRs contain a larger number of time series. In this figure, we choose Bundle 1, which represents the rather quieter taxi drop off zones in Manhattan, as the number of drop-offs there is rarely over 60 during any hour of the week. In this example, there is also a clear daily routine in all bundles, with the drop-offs reaching a local maximum twice per day, suggesting the rush hours in New York City, when people commute to and from their work. In almost all bundles, the daily pattern is significantly different on Saturdays and Sundays, which confirms the intuition that during weekends people do not tend to commute in a routinely fashion. Overall, such visual representations of information digested from massive time series data can easily catch users’ attention to important phenomena and ongoing trends, confirming the usefulness of our approach.

5.3 Performance Results

In order to evaluate the performance of our approach on larger datasets, we built the index using the synthetic dataset and examined its response time for different zoom levels on the map. Since this is intended as an interactive application, where the summarization method is triggered as soon as the user moves the map, response times must be adequately small. Ideally, the response time should be in the order of milliseconds. In our method, this is facilitated by the fact that the search along a path stops once it encounters a node whose MBR is contained in the actual map extent (rectangle). In this experiment, we measure the response time for different zoom levels, since zooming-in requires deeper traversal of the BTSR-tree index in order to locate the relevant nodes. We use map scales to indicate the different zoom levels.

Figure 5 depicts traversal costs for different map scales over the areas covered by the three datasets. More specifically, the water and synthetic datasets cover the area of the city of Alicante, Spain, whereas the taxi dataset covers the wider metropolitan area of New York City. Response time in all cases is equal or lower than one second, which makes this method suitable for interactive visualization. The synthetic dataset, due to its very high density is significantly slower than the rest, however still the results are

obtained in less than a second. The response for the water dataset is almost instant due to its small size and very low density.

Initially, in all cases, at the largest scale, the visible area of the map contains all the time series in the dataset, thus it only has to retrieve information from the root of the index. Then, as we zoom in, more nodes have to be visited, as the MBRs of the accessed nodes begin to overlap with the map rectangle and their children have to be retrieved. The worst case for the synthetic dataset is at scale 1:5000, which roughly corresponds to a large neighborhood of the city, where many time series are located. For the taxi dataset, the worst case is at 1:20000, which corresponds to the wider Manhattan area and then the response time gradually drops due to the lower dataset density. The number of nodes accessed in each case is proportional to the response times, ranging from...
6 CONCLUSIONS AND FUTURE WORK

In this paper, we introduced a method for map-based visual exploration of geolocated time series data. To that end, we proposed a summarization approach over geolocated time series, which allows a visual analytics application to retrieve the required information. Such retrieval can be achieved at low latency, thus being suitable for interactive exploration of large volumes of such data. The results can be displayed on a map, depicting the relevant MBRs and the number of time series contained in each one, for a selected pattern detected in the time series data. Thanks to the support of a robust hybrid indexing technique, the patterns detected at a given zoom level are calculated via k-means clustering over the time series that reside in the currently visible part of the map. Our experiments on a large-scale synthetic dataset indicated that the visualization can be rendered adequately fast for use in interactive map-based applications. Additionally, we presented indicative demonstrations of the visualizations generated on two real-world datasets from different domains, confirming that these visualizations are helpful in revealing patterns both on the time series themselves as well as their geographic distribution.

Our ongoing and future work focuses on supporting more detailed visual analytics and identifying more fine-grained patterns through visual exploration. One possible extension would be to enable zooms along time, so that the user can identify patterns and their spatial distribution, not only over the entire time series, but also over particular intervals. Further, it would be interesting to drill-down in a particular summarized result and discover whether there are differentiation in the spatial distributions of its constituent, more detailed patterns.
An Interactive 3D Visualization for the LOD Cloud
Maria-Evangelia Papadaki, Panagiotis Papadakos, Michalis Mountantonakis, Yannis Tzitzikas
Institute of Computer Science, FORTH-ICS, GREECE, and
Computer Science Department, University of Crete, GREECE
{marpap,papadako,mountant,tzitzik}@ics.forth.gr

ABSTRACT
The LOD (Linked Open Data) cloud currently contains thousands of published datasets. Existing visualizations, like the Linking Open Data cloud diagram, are useful for getting an overview of its size, the datasets and their connectivity. An interesting question is whether we could come up with more informative and more interactive visualizations that could make evident more features of the datasets for aiding the inspection and the discovery of related datasets. To this end we propose an interactive 3D visualization that adopts the metaphor of urban area. In brief, each dataset is visualized as a building, whose features (e.g. volume) reflect various dataset’s features (e.g. number of triples), while the proximity of the buildings (and other features) indicates the commonalities of the datasets. The introduced approach supports various shapes of buildings and various placement algorithms: mountainside, orthogonal spiral, concentric spiral, and similarity-based adaptations of force-directed algorithms. The visualization is interactive, i.e. it allows the user to zoom in any part of the model, to change the perspective, to change the shape of the buildings and their placement, to see all the connections or only those of one dataset, and others. The paper details the construction process and provides examples over real datasets including the entire LOD cloud, and describes the pros and cons of each layout.

KEYWORDS
Linked Data, Connectivity of Linked datasets, Interactive3D Visualization

1 INTRODUCTION
During the last years we observe an increasing trend towards publishing data as LOD. Thousands of datasets have been published and various visualizations that give an overview of their number and interconnections have been proposed (e.g. see [1, 7, 8]). The classical visualization of the LOD cloud (Figure 1(left)), depicts each dataset as a circle (whose size indicates the size of the dataset in triples). The commonalities between two datasets (in terms of common URIs) are made evident by an edge that connects the dataset’s circles. Such visualizations are useful for getting an overview of the entire LOD cloud, or for a part of it, or for a particular set of RDF datasets. There are various visualization-driven tasks. In our work we focus mainly on tasks related to datasets inspection, datasets monitoring, dataset selection and navigation across multiple linked datasets. The basic question we address here, is: can we come up with visualizations of the LOD cloud which are more informative (i.e. which can make evident more “features” of the datasets) and are easily conceivable? Based on this motivation, in this paper we propose an interactive 3D visualization that adopts a quite familiar metaphor, specifically that of an urban area where each dataset is visualized as a building. An indicative screenshot of the LOD cloud according to the interactive 3D visualization that we propose, is shown in Figure 1(right). In a nutshell the contributions of this paper are: (i) it introduces and motivates a novel interactive 3D model for LOD datasets that adopts the metaphor of urban area, (ii) it introduces several variations of the model, and discusses the pros and cons of each one, and (iii) it demonstrates the application of the model over the datasets of each domain (government, media, etc.) and the entire LOD cloud. The rest of this paper is organized as follows: §2 describes the context, §3 describes the main components of the interactive 3D model, and its application, §4 describes the implementation of the visualization system as well as directions that are worth further work and research, and finally §5 concludes the paper. A running prototype is already available to the community and it is accessible through http://www.ics.forth.gr/isl/3DLod (needs a recent web browser supporting WebGL).

2 CONTEXT
Visualization has been recognized as important for dataset discovery and dataset selection [10], which consist two of the most emerging challenges for the web of data [5, 9]. A number of visualization approaches and tools for Linked Data have been proposed, some indicative of which are described in [6]. The most widely known visualization diagram of the LOD is the 2D Linking Open Data cloud diagram, which consists of datasets that have been published in Linked Data format by contributors to the Linking Open Data community project and other individuals and organisations. It is based on metadata collected and curated by contributors to the datahub.io as well as on metadata extracted from periodic crawls of the Linked Data web. The 2014 crawled version of the diagram is shown in Figure 1(left). We refer to the Linking Open Data cloud that was available from 2014-08-30 to 2017-01-251 that contains datasets from the following nine domains (in parenthesis the percentages of datasets that fall in each category): government (23.85%), publications (23.33%), social web (15.78%), life sciences (11.05%), cross-domain (7.19%), user-generated content (7.36%), geographic (4.21%), media (3.68%), and linguistics (3.50%). The size of the circles corresponds to the number of triples in each dataset. Only five sizes of circles (very large, large, medium, small, very small) are supported each corresponding to a particular size interval (> 1 B, 10M-1B, 500K-10M, 10K-500K, < 10K resp.). The arrows between two circles indicate the existence of at least 50 links between the corresponding two datasets. A link is considered as an RDF triple where subject and object URIs are in the namespaces of different datasets, while the direction of the arrows indicates the dataset that contains the links. The thickness of the arrow corresponds to the number of links. Three levels of thickness are supported (thin, medium, thick) each corresponding to one interval ((0, 1K), [1K, 100K] and [100K, ∞) respectively). Finally,

1 Accessible through http://lod-cloud.net/versions/2014-08-30/lod-cloud_colored.svg
each circle is colored differently for indicating the 9 different domains of the datasets. A new version of the Linking Open Data cloud diagram was released on 2017-01-26.\textsuperscript{2} That version contains almost double the number of datasets (i.e. 1163). Datasets are again visualized as circles however only three sizes of circles (large, medium, small) are supported. The links are interactive and their direction is indicated through color. However, the clustering of the datasets is not favorable in all cases and the labels are less readable in comparison to the 2014-2017 version of the diagram.

3 AN INTERACTIVE URBAN 3D VISUALIZATION FOR LOD DATASETS

3.1 Dataset Notations

Let $S = S_1, \ldots, S_k$ be the set of datasets. Each dataset $S_i$ consists of a set of triples (i.e. a set of subject-predicate-object statements), denoted by $\text{triples}(S_i)$. We shall use $U_i$ to denote the URIs, $L_i$ to denote the literals and $BN_i$ to denote the blank nodes that appear in $\text{triples}(S_i)$. Hereafter, we consider only those URIs that appear as subjects or objects in a triple as our primary focus is on the data (not on schema). The number of common URIs between two datasets $S_i$ and $S_j$, is given by $|U_i \cap U_j|$. We define the Links between two datasets as follows: $\text{Links}_{i,j} = U_i \cap U_j$.

If $T$ is a set of triples, then we can define the degree of a URI $e$ in $T$ as $\text{deg}_T(e) = |\{(s,p,o) \in T \mid s = e \text{ or } o = e\}|$. While for a set of URIs $E$ we can define their average degree in $T$ as $\text{deg}_T(E) = \text{avg}_{e \in E}(\text{deg}_T(e))$. Now for each dataset $S_i$ we can compute the average degree of the elements in $U_i$ by considering $\text{triples}(S_i)$, i.e.: $\text{Deg}(U_i) = \text{avg}_{e \in U_i}(\text{deg}_\text{triples}(S_i)(e))$.

3.2 Buildings Representation

The main idea is that we visualize each dataset $S_i$ as a building $b_i$. The volume of each building represents the number of triples of the respective dataset ($|\text{triples}(S_i)|$). As regards the types of the buildings, we support the following options: (a) cubes, (b) "context"-dependent cuboids, and (c) "feature"-based cuboids.

In (a), each dataset $S_i$ is represented by a cube with edge length equal to $\sqrt{|\text{triples}(S_i)|}$.

In (b) we use "context-dependent" cuboids. The footprint of the buildings is computed based on either the biggest dataset ($b_{\text{Big}}$ mode) or the smallest dataset ($b_{\text{Small}}$ mode). In the $b_{\text{Big}}$ mode the building of the biggest dataset is a cube, while in the $b_{\text{Small}}$ mode the cube corresponds to the smallest dataset. Consequently, the buildings of the datasets that have enough triples tend to become skyscrapers.

In (c), i.e. "feature-based" cuboids, the shape depends on the features of the corresponding datasets. Since $|\text{triples}(S_i)| \approx (|U_i| + |L_i| + |BN_i|) \cdot \text{Deg}(U_i)$, the height of the building is set to be analogous to $|U_i| + |L_i| + |BN_i|$, and the footprint of the building analogous to $\text{Deg}(U_i)$. Specifically, assuming square footprints, we have $\text{height}(b_i) = |U_i| + |L_i| + |BN_i|$ and $\text{width}(b_i) = \sqrt{\text{Deg}(U_i)}$. The volume of the building $b_i$ consists of $\text{triples}(S_i)$; if its degree is low it will become a high building with a small footprint, whereas if its degree is high then the building will be less tall but will have a big footprint.

For getting building sizes that resemble those of a real urban area, a calibration is required. For this reason we introduce an additional parameter $F$, through which we can obtain the desired average ratio of height/width of the buildings. Specifically, let $r$ be the desired ratio (e.g. 3 for three-floor buildings) provided by the user. We can add a parameter $F$ to the definition of $\text{height}$ and $\text{width}$:

$$\text{height}(b_i) = (|U_i| + |L_i| + |BN_i|)/F$$

$$\text{width}(b_i) = \sqrt{\text{Deg}(U_i)/F}.$$ 

3.3 Placement of the Buildings

Below we describe four different building layout approaches, that our system supports.

1. Mountainside Layout. The $k$ buildings are placed in an orthogonal $\sqrt{k} \times \sqrt{k}$ grid. The biggest building is placed in one edge of the square area. The second bigger is placed next to the first, and so on, until reaching the end of a row, where it continues the same procedure in the next one until there are no

\textsuperscript{2} by Andrejs Abele, John P. McCrae, Paul Buitelaar, Anja Jentzsch and Richard Cyganiak, http://lod-cloud.net/
Table 1: Comparison of visualizations for Linked Datasets

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Mountain Side</th>
<th>Orthogonal Spiral</th>
<th>Cyclic Spiral</th>
<th>Similarity-based layout</th>
<th>LOD Cloud Diagram (2D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accurate size</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>only 5 sizes</td>
</tr>
<tr>
<td>Features (e.g., Degree)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Connectivity</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Interactive</td>
<td>rich</td>
<td>rich</td>
<td>rich</td>
<td>rich</td>
<td>poor (only 2D with zoom)</td>
</tr>
<tr>
<td>Time Complexity</td>
<td>O(n)</td>
<td>O(n)</td>
<td>O(n)</td>
<td>O(e + n^2)</td>
<td>not mentioned</td>
</tr>
<tr>
<td>Distinctive characteristic</td>
<td>Readability</td>
<td>Fast overview</td>
<td>Effective space exploitation even for power law distributed datasets</td>
<td>Focus on connectivity (less edge crossings)</td>
<td>Label readability, connections, clustered datasets by domain</td>
</tr>
</tbody>
</table>

(b) a spiral-like placement seems beneficial as it would result to a round coverage of the space,

(c) collisions should never occur,

(d) no big empty spaces, especially in the outer area that hosts the majority of the buildings which are small.

For the above requirements, we devised a new 2D placement algorithm called Concetric Spiral. The buildings, in an descending order with respect to their size, are placed on concentric rings. The radius of the first (smallest) ring is the size of the biggest building. The placement of the subsequent buildings is done as follows. We compute the minimum chord that is required to avoid collisions based on the sizes of the current and the previous building. Then we compute the corresponding angle, and we place the new building at the corresponding spot of the circle. The sought angle is \( \theta = 2 \arcsin \frac{r}{k} \). Just before we reach \( 2\pi \), we start the next bigger ring whose radius, is the radius of the previous ring increased by the size of the last drawn building (plus a number accounting for "roads"). In this way the concentric rings become denser as the buildings get smaller avoiding the unnecessary empty spaces. The algorithm is appropriate for sets of datasets whose sizes vary a lot, even if they exhibit a power law distribution (i.e. very few big datasets and too many small ones, see [3,9] for measurements about current RDF datasets). A screenshot of the layout based on Cyclic Spiral is shown in Figure 1 - right and Figure 2 - upper right. The algorithm has \( O(n) \) time complexity (\( n \) is the number of buildings).

4. Similarity-based layout. According to this algorithm, the more commonalities two sources have (common URIs, common literals, owl:sameAs relationships, etc.) the closer the corresponding buildings are placed. One way to specify the location of each building is to adopt a force-directed placement algorithm. In our case, we have modified the Fruchterman-Reingold force directed algorithm [4] as adapted to three \( \beta \). This algorithm satisfies the following two principles: a) vertices connected by an edge should be drawn near each other and b) vertices should not be drawn too close to each other. Figure 3 shows an indicative layout produced by the similarity-based algorithm.

Comparison. Table 1 summarizes the distinctive characteristics of each visualization approach including the 2D LOD Cloud diagram. The value "rich" in the line "interactive" refers to interactive selection, zooming, panning, rotation, and control of visibility of labels and connections.

3.4 Visualizing the Links of Datasets

If there are links between two datasets \( S_i \) and \( S_j \) then a line segment is created, resembling a road that connects the corresponding buildings (see the left side of Figure 4). The links can be also visualized as bridges (see the right side of Figure 4). The width of these bridges/roads, indicates the strength of the connection that

\[^1\](https://github.com/davidpiegza/Graph-Visualization)
the correlated datasets have, and it is calculated by the division of the number of links between $S_i$ and $S_j$ with the number of links of the most connected pair (i.e., maxLinks): $width(i,j) = \frac{Links(i,j)}{\max(Links)}$.

### 3.5 Application Cases

We downloaded manually 287 RDF datasets including their content (i.e., triples, URIs, etc.) from the following resources: (a) the dump of the data which were used in [11], (b) online datasets from datahub.io website and (c) a subset of DBpedia version 3.9. To test the algorithms in even bigger datasets we managed to find metadata from datahub.io for 600 datasets of various domains. Comparing to the 287 datasets (see Figure 1(right)), for most of these 600 datasets we were not able to access and download their content (i.e., triples, URIs, etc.). However, we managed to find some basic metadata for these datasets in datahub.io. Unfortunately, in datahub.io there is a lack of information for other features of these datasets such as the number of URIs, literals, blank nodes and degree of URIs. Therefore, it is not possible to produce feature-based buildings for these datasets, although the proposed visualizations can support feature-based buildings for thousands of datasets. Figure 5 shows on the left side the cyclic spiral layout and on the right side the orthogonal layout for this set of 600 datasets.

![Figure 5: 3D Visualizations of 600 datasets](image)

### 4 IMPLEMENTATION AND FUTURE STEPS

We have implemented a web-based visualization system, which could be easily accessible by any user. We used the JavaScript library Three.js which in turn uses the WebGL API, which is widely supported by all modern desktop and mobile browsers without the use of plugins. Three.js offers a less tedious programming environment in comparison to WebGL, by abstracting away many of the WebGL details, which is a JavaScript API that allows the creation of GPU accelerated 3D graphics and animations inside the environment of a web browser [2].

Figure 6 shows an overview of the web-based visualization system. The visualization is interactive, allowing the user to zoom in any part of the model. For instance, one can change the perspective, the shape of the buildings or their placement, search for a dataset through an auto-completion search, see all the connections or those of one dataset, and others. The presented model could be improved in several ways. Below we sketch two indicative enrichments: (a) for aiding the user to get a more informative and “live” overview immediately the system could be enriched with “guided tours”, i.e. with trails of camera movements over the space occupied by the buildings and (b) for reducing the crossings of the edges, each set of buildings that forms a strongly connected component could be visualized as a small round park (or roundabout) where only one line segment connects each building to that park.

![Figure 6: Overview of the web GUI of the system](image)

### 5 CONCLUDING REMARKS

The proposed 3D interactive system: (i) illustrates accurately the relative sizes of the datasets in triples, (ii) can indicate the average degree of the datasets, (iii) allows the user to control which connections to show or hide, (iv) makes evident (through the layout algorithms) the differences in the sizes of datasets or their commonalities. It supports various building types (cubes, context-dependent cuboids, feature-based cuboids), as well as several layout algorithms (mountainside, orthogonal spiral, cyclic Spiral, similarity-based adaptations of force-directed algorithms), that order the buildings appropriately, depending on the user needs, and similarity-based adaptations of force-directed algorithms.

Acknowledgements. Work partially supported by a) the EU project BlueBRIDGE (Building Research environments for fostering Innovation, Decision making, Governance and Education to support Blue growth), H2020-EINFRA-2015-1, 2015-2016 and b) the General Secretariat for Research and Technology (GSRT) and the Hellenic Foundation for Research and Innovation (HFRF).

### REFERENCES


Topic-aware Network Visualisation to Explore Large Email Corpora

Tim Repke
Hasso Plattner Institute
Potsdam, Germany
tim.repke@hpi.de

Ralf Krestel
Hasso Plattner Institute
Potsdam, Germany
ralf.krestel@hpi.de

ABSTRACT
Nowadays, more and more large datasets exhibit an intrinsic graph structure. While there exist special graph databases to handle even increasing amounts of nodes and edges, visualising this data becomes infeasible quickly with growing data. In addition, looking at its structure is not sufficient to get an overview of a graph dataset. Indeed, visualising additional information about nodes or edges without cluttering the screen is essential. In this paper, we propose an interactive visualisation for social networks that positions individuals (nodes) on a two-dimensional canvas such that communities defined by social links (edges) are easily recognisable. Furthermore, we visualise topical relatedness between individuals by analysing information about social links, in our case email communication. To this end, we utilise document embeddings, which project the content of an email message into a high dimensional semantic space and graph embeddings, which project nodes in a network graph into a latent space reflecting their relatedness.

1 INTRODUCTION
In our modern information society we produce substantial amounts of data each day. A large portion of it comes from the communication on social media platforms or through emails. Special graph databases enable the efficient storage of these large communication networks and provide interfaces to query or analyse the data. Visualising networks in their entirety on the other hand is a very challenging task. Users investigating a communication network want to find information about whom does who communicate with whom about what. These kind of networks can be found in many different shapes. Modern social networks, such as Twitter or Facebook exhibit similar structures as classic, online social networks [20]. We investigate another type of social network: a collection of emails.

Given the communication data over a year or more, it is practically impossible to gain an overview or quick insights into the latent network structure with a basic approach as shown in 1. Also, in such a traditional network visualisation, information about the content of messages sent between individuals is lost. Besides these traditional systems, more exotic approaches use the metaphor of geographical maps [17] to visualise networks, for example using topology to reflect connectivity of densely connected social communities. The map analogy can also be used to visualise the contents of documents by embedding them into a high dimensional semantic space [15] and projecting it on the map as a document landscape. In order to highlight how relationships form and change based on the interactions, the metaphor of a growing tree can be used (ContactTrees [18]). Although this reflects temporal aspects of dynamic networks well, it focuses on one person as the root, thus an overview of the entire network is lost. CactusTrees [6] on the other hand represent hierarchical structures with the goal of untangling overlaid bundles of intersecting edges, making distant connections more apparent. As higher order dependencies may get lost in traditional visualisations, HoNVis [21] adds nodes to encode dependencies in chains of interactions. Usually, a communication network has many nodes and overlapping connections already, so Yang et al. [23] rather focus on discovering overlapping cores to improve the identification of community boundaries to highlight global latent structures. Similarly, Gronemann et al. [11] use the metaphor of islands and hills to visualise clustered graphs, making densely connected communities clearly noticeable. The edges are bundled and follow valleys of the resulting topology, thus making relationships between other communities hard to follow. MapSets [7] assume a graph that was laid out using embeddings reflecting communities. An algorithm then draws regions around clusters of nodes, such that the bounding shapes are contiguous and non-overlapping, but yet abstract. Another approach to visualise networks at full scale is to aggregate nodes based on their spatial distribution and thereby allowing for a simple exploration with contour lines and heatmap overlays to emphasise latent structures as proposed by Hildenbrand et al. [13].

Document visualisation aims to visualise the content, such that users gain quick insights into topics, latent phrases, or trends. Tiara [22] extracts topics and derives time-sensitive keywords to depict evolving subjects over time as stacked plots. Other approaches project documents into a latent space, using topic models or embeddings. Creating scatter-plots of embedded documents of a large corpus may result in a very dense and unclear layout, so Chen et al. [4] developed an algorithm to reduce overfull visualisations by picking representative documents. A different approach is taken by Fortuna et al. [8], who do not show documents directly, but generate a heatmap of the populated canvas and overlay it with salient phrases at more densely populated areas from the underlying documents in that region. Friedl et al. [10] extend that concept by drawing clear lines between regions and colouring them. They also add edges between salient phrases based on co-occurrences in the texts. Most recently Cartograph [19] was proposed, which is visually very similar to previous approaches, but uses pre-rendered information of different resolution and map technology to enable a responsive interactive visualisation. Regions are coloured based on underlying ontologies from a knowledge-base.

Our goal is to merge approaches for network and document visualisations in one interactive user interface. This means to integrate multiple dimensions of email datasets including time, interactions, users and topics into a 2D map representation. Giving an overview over latent structures and topics in one map may significantly improve the exploration of a corpus by users.
unfamiliar with the domain and terminology. Also domain experts could benefit from such an overview, e.g. by easily being able to identify global patterns in the data.

A specific application scenario that could benefit from such integrated, interactive visualisations is the analysis of large, unstructured, heterogeneous data collections. Data-driven journalism [5] often has to deal with leaked, unstructured, very heterogeneous data, e.g. in the context of the Panama Papers, where journalists needed to untangle and order huge amounts of information, search entities, and visualise found patterns [3]. Similar datasets are of interest in the context of computational forensics [9]. Auditing firms and law enforcement need to sift through huge amounts of data to gather evidence of criminal activity, often involving communication networks and documents [14].

2 INTERACTIVE VISUALISATION

Systems for document exploration largely vary in what they display and how users interact with them. This depends partly on the available raw data, but also on information extracted from pre-processing or enrichment with external sources. Figure 1 shows a basic visualisation of the network graph extracted from an email corpus. Although it is an improvement over only listing connections, large densely connected graphs quickly become hard to read and information about the email contents is lost.

Exploring document collections can be seen as a top-down approach, where the system provides abstract overviews of the entire document collection and users incrementally refine the search, narrowing the results to just a few documents of interest. Such a top-down approach may help users without prior knowledge to get a sense for the data by visualising high level latent structures of communication networks or the topical distributions.

In the scope of this work we primarily consider documents to be emails or data attached to them. The sender, recipients, time, and content can directly be extracted from the raw data. We call these – and results from further processing – dimensions that can be visualised. From the contents one may infer named entities, topics, embeddings, or salient phrases, while the communication network spanned by sender-recipient pairs can be used to detect salient structures and hierarchies. The temporal information enables the previously mentioned data to be analysed over time to detect evolving or changing patterns.

There are numerous ways to visualise each dimension on its own or in combination with others. The requirement of a dimension and its priority in a visualisation is dictated by the system objective. From the wide range of possibilities, we strive for a system which supports the exploration of a large collection of documents without any prior knowledge about its content and individuals involved.

In our system, we use the names and email addresses of senders and recipients (individuals), communication network, semantic vector representations of email contents, and as part of an overlay the timestamps of emails and propose a graph layout over a document landscape that visually describes who talks with whom about what at a given time period.

3 SYSTEM ARCHITECTURE

Visualising communication networks in a topic-aware fashion to explore documents and salient structures is not straightforward. Different layout objectives may produce contradicting results and the challenges of processing big data need to be addressed [2]. In this section, we describe algorithmic approaches behind the system we are working on. For a discussion of engineering aspects on how to store, serve, and render the map-like data, we refer to the Cartograph stack [19], as we will focus on the process how to get the information that the map is generated from.

We visualise the embedded emails as dots in a two-dimensional landscape in which individuals are placed as nodes connected by edges. All emails between two individuals are reduced into one edge reducing visual complexity and making it easier to detect salient structures. However, that comes with the trade-off that nodes and edges cannot be perfectly placed in the landscape to cover all semantic aspects of the communication between them, but rather an estimate. Our very early prototype placed some individuals with no dominant topic in a crowded area in the centre of the landscape as shown in 2, where colours of opaque dots for emails correspond to that of the sender. Although the network visualisation at this point does not make connections more clear than in 1, users can already distinguish individuals with similar or unrelated topics.

Our proposed algorithm to find a stable network layout has three stages, namely an (i) initialisation phase which creates the landscape and roughly places nodes and connections, an (ii) update phase which iteratively updates the node placement towards a better fit, and finally a (iii) post processing phase where edges become splines to make latent structures more clear and a map topology is added.

Initialising the Landscape. To generate the document landscape, we first process the network graph to roughly determine regions, where documents will be placed. Therefore we apply node2vec [12] to the communication network and embed each individual’s node. We separate the graph into communities $P_i \subset P$ using the kernel density of the resulting populated space at
threshold \( \kappa \), where a higher \( \kappa \) results in more, but smaller communities. For each community \( P_i \), pairwise neighbourhood similarities are calculated using euclidean distance between nodes, forming the triangular matrix \( S_i \), where \( s_{kl} \) is the similarity between \( p_k, p_l \in P_i \).

Furthermore, we train document embeddings [15, 25] on all emails and use them to infer high dimensional semantic vector representations. Let \( M_i \) be the set of emails that originated in community \( P_i \). For each email \( m \in M_i \), the dimensionality is reduced using t-SNE [16], which retains possible semantic clusterings of documents in the higher dimensional space. The resulting two-dimensional vectors are then placed as dots on the map using the centre of embedded network communities as the respective origin, whereas the size is determined by the number of related individuals.

We also initialise communication network’s layout. Thereby, the starting position of a node representing an individual is determined by the normalised sum of two-dimensional vectors of all emails he or she has sent or received. This way, we implicitly group semantically related individuals into communities as frequent communication biases this normalised sum. Straight edges are added between the nodes if the respective individuals exchanged emails. Note, that many edges may only represent a small number of emails. Applying a variable threshold \( \sigma \) can reduce the computational load in later stages, as these edges will not impact the overall layout very much. They can be added again as the user requests a detailed visualisation by zooming in or through other interactions.

In the algorithm’s second stage, we iteratively try to improve the layout of the communication network by finding a balance between the closeness of nodes to semantic context and densely connected neighbourhoods a node belongs to. Therefore, for each individual \( p_j \in P \) we use linear regression to fit a line \( \bar{m}_p \) though all two-dimensional vectors of emails he or she has sent or received. As a node is placed near this line, it remains in a semantically good position.

Adjusting the Network Layout. The first stage of our proposed algorithm produces a fixed document landscape and roughly fits the communication network on top. We now aim to incrementally adapt the layout of the graph to better reflect salient structures in the network while keeping each individual’s node close to the reflective semantic area in the landscape.

Therefore we define a score quantifying how well the current layout fits these objectives:

\[
\sum_{p_i \in P} \left( \sum_{p_j \in P} s_{ij} d(p_i, \bar{m}_p) + \sum_{p_k \in P} \theta \left( s_{ij} - d(p_i, p_j) \right) \right)
\]  

where \( d(\cdot, \cdot) \) is the distance between two nodes (zero if no connection exists) or shortest distance from a node to its ideal line. To adjust the layout towards either a better semantic or structural fit, we introduce parameters \( \theta \) and \( \eta \).

In order to minimise 1, we use stochastic gradient descent. In each iteration step, we can derive the direction and magnitude each node should be moved towards a better semantic fit and closer proximity to it’s neighbourhood in the network.

The semantic gradient for \( p_j \in P \) is defined by

\[
\delta_j^s := (\bar{m}_p - p_j) \frac{\| \bar{m}_p - p_j \|}{\| \bar{m}_p - p_j \|}
\]

where \( \bar{m}_p \) is the closest point on \( \bar{m}_p \) to \( p_j \) and \( \| \| \) denotes the euclidean norm, while neighbourhood gradient is defined by

\[
\delta_j^n := \sum_{p_k \in P \setminus \{p_j\}} \left( \frac{\| p_j - p_k \|}{\| p_j - p_k \|} - s_{jk} \right)
\]

where \( p_j \sim p_k \) denotes that an edge exists between \( p_j \) and \( p_k \).

With the definitions in 2 and 3, we can formulate the update vector \( \delta_j \) for node \( p_j \) in \( P \) as

\[
\delta_j = \xi \left( \delta_j^s + \eta \delta_j^n \right)
\]

where \( \xi \) is the learning rate and \( \theta, \eta \) as before parameters to weight between a better semantic or neighbourhood fit.

Most likely, complex network structures might prevent the stochastic gradient descent to find a stable minimum, so the score of the objective function should be monitored or intermediate layouts be visually evaluated to determine a satisfactory result.

Post Processing. Lastly, we use the post processing stage to enhance the readability of our visualisation. Densely connected communities in the graph are potentially hard to read, thus we apply edge bundling [1] to visually clear latent structures. We also apply MapSets [7] to separate the regions for each community. Since semantically similar emails may appear in different communities, we apply colouring based on clusters in the original global document embedding space to retain this aspect. Choosing the colours depends on the number of latent topics that should be depicted [24]. If the topic number exceeds 25-30 topics, grouping topics and allowing for zooming within a two-level topic-hierarchy ensures distinguishable colors for up to 10 subtopics (25 \( \times \) 10 = 25). In order to represent temporal aspects of the data, we calculate the kernel density of the document landscape for fixed time-intervals, which can be used to add heat-map overlays that users can select later on.
4 CONCLUSION AND VISION

In this paper, we described an algorithm to lay out a communication network on top of a landscape of semantically embedded emails. This is still work in progress, thus 3 shows only a manually drawn mock-up of the visualisation we envision. In it, individuals are represented as nodes positioned such that densely connected communities are visually clustered. Edges describe the email traffic, where the opacity and thickness is used to indicate the frequency of messages between the nodes they connect.

The semantic representations of emails are used to place dots on a background layer which we call the document landscape. This landscape is used as additional input to the graph layout algorithm, aiming to place a node within corresponding semantic regions. The colouring of regions in the landscape is derived from densely connected communities in the communication graph. Optionally, representative words are selected for densely populated areas in the landscape, so that users get a rough idea about subjects in that area. The aforementioned timestamps of emails can be used to generate a heatmap overlay to show the activity in a certain time interval which is controlled by a slider. Similar to modern geographical maps, zooming into a region reveals more details. In our case, less prominent individuals and their connections are shown along with additional salient phrases from the document landscape. Selecting a node will not only highlight connected edges but may also temporarily show more edges which were previously hidden at that zoom level. The user will also be able to retrieve documents with the help of a selection rectangle or clicking dots in the document landscape.

In future work, we hope to evaluate this system using full-scale real-world data as well as practitioners from journalism and auditing. It may also be interesting to experiment with embedding methods, which take both the emails and the network graph as input and directly project the inferred representations into the two-dimensional landscape to simplify the proposed algorithm.

REFERENCES


Big Mobility Data Analytics

BMDA

Nikos Peleakis, University of Piraeus, Greece
Gennady Andrienko, Fraunhofer IAIS, Germany
Alexander Artikis, University of Piraeus & NCSR Demokritos, Greece
Fosca Giannotti, ISTI CNR Pisa, Italy
Akrivi Kiousi, Intrasoft International, Greece
Themis Palpanas, Paris Descartes University, France
Yannis Theodoridis, University of Piraeus & Athena RIC, Greece
A Distributed Online Learning Approach for Pattern Prediction over Movement Event Streams with Apache Flink

Ehab Qadah
Fraunhofer IAIS
Sankt Augustin, Germany
Ehab.Qadah@iais.fraunhofer.de

Michael Mock
Fraunhofer IAIS
Sankt Augustin, Germany
Michael.Mock@iais.fraunhofer.de

Elias Alevizos
NCSR ‘Demokritos’
Athens, Greece
alevizos.elias@iit.demokritos.gr

ABSTRACT
In this paper, we present a distributed online prediction system for user-defined patterns over multiple massive streams of movement events, built using the general purpose stream processing framework Apache Flink. The proposed approach is based on combining probabilistic event pattern prediction models on multiple predictor nodes with a distributed online learning protocol in order to continuously learn the parameters of a global prediction model and share them among the predictors in a communication-efficient way. Our approach enables the collaborative learning between the predictors (i.e., “learn from each other”), thus the learning rate is accelerated with less data for each predictor. The underlying model provides online predictions about when a pattern (i.e., a regular expression over the event types) is expected to be completed within each event stream. We describe the distributed architecture of the proposed system, its implementation in Flink, and present experimental results over real-world event streams related to trajectories of moving vessels.

1 INTRODUCTION
In recent years, technological advances have led to a growing availability of massive amounts of continuous streaming data (i.e., data streams observing events) in many application domains such as social networks [23], Internet of Things (IoT) [24], and maritime surveillance [31]. The ability to detect and predict the full matches of a pattern of interest (e.g., a certain sequence of events), defined by a domain expert, is typically important for operational decision making tasks in the respective domains.

An event stream is an unbounded collection of time-ordered data observations in the form of a tuple of attributes that is composed of a value from finite event types along with other categorical and numerical attributes. In this work, we deal with movement event streams. For instance, in the context of maritime surveillance the event stream of a moving vessel consists of spatio-temporal and kinematic information along with the vessel’s identification and its trajectory related events, based on the automatic identification system (AIS) [28] messages that are continuously sent by the vessel. Therefore, leveraging event patterns prediction over real-time streams of moving vessels is useful to alert maritime operation managers about suspicious activities (e.g., fast sailing vessels near ports, or illegal fishing) before they happen. However, processing real-time streaming data with low latency is challenging, since data streams are large and distributed in nature and continuously arrive at a high rate.

In this paper, we present the design and implementation of an online, distributed and scalable pattern prediction system over multiple, massive streams of events. More precisely, we consider event streams related to trajectories of moving objects (i.e., vessels). The proposed approach is based on a novel method that combines a distributed online prediction protocol [8, 16] with an event forecasting method based on Markov chains [2]. It is implemented on top of the Big Data framework for stream processing Apache Flink [13]. We evaluate our proposed system over real-world data streams of moving vessels, which are provided in the context of the daAcron project.[1]

The rest of the paper is organized as follows. We discuss the related work and used frameworks in Section 2. In Section 3, we describe the problem of pattern prediction, our proposed approach, and the architecture of our system. The implementation details on top of Flink are presented in Section 4 and the experimental results in Section 5. We conclude in Section 6.

2 RELATED WORK AND BACKGROUND
2.1 Related work

Pattern prediction over event streams. The task of forecasting over time-evolving streams of data can be formulated in various ways and with varying assumptions. One common way to formalize this task is to assume that the stream is a time-series of numerical values, and the goal is to forecast at each time point \( n \) the values at some future points \( n+1, n+2, \ldots \), (or even the output of some function of future values). This is the task of time-series forecasting [25]. Another way to formalize this task is to view streams as sequences of events, i.e., tuples with multiple, possibly categorical, attributes, like \( event\ type,\ timestamp \), etc., and the goal is to predict future events or patterns of events. In this paper, we focus on this latter definition of forecasting (event pattern forecasting).

A substantial body of work on event forecasting comes from the field of temporal pattern mining where events are defined as 2-tuples of the form \( (Event\ Type,\ Timestamp) \). The ultimate goal is to extract patterns of events in the form either of association rules [1] or frequent episode rules [22]. These methods have been extended in order to be able to learn not only rules for detecting event patterns but also rules for predicting events. For example, in [32], a variant of association rule mining is where the goal is to extract sets of event types that frequently lead to a rare, target event within a temporal window.


© 2018 Copyright held by the owner/author(s). Published in the Workshop Proceedings of the EDBT/ICDT 2018 Joint Conference (March 26, 2018, Vienna, Austria) on CEUR-WS.org (ISSN 1613-0073). Distribution of this paper is permitted under the terms of the Creative Commons license CC-by-nc-nd 4.0.
In [19], a probabilistic model is presented for calculating the probability of the immediately next event in the stream. This is achieved by using standard frequent episode discovery algorithms and combining them with Hidden Markov Models and mixture models. The framework of episode rules is employed in [9] as well. The output of the proposed algorithms is a set of predictive rules whose antecedent is minimal (in number of events) and temporally distant from the consequent. In [35] a set of algorithms is proposed that target batch online mining of sequential patterns, without maintaining exact frequency counts. As the stream is consumed, the learned patterns can be used to test whether a prefix matches the last events seen in the stream, indicating a possibility of occurrence for events that belong to the suffix of the rule.

Event forecasting has also attracted some attention from the filed of Complex Event Processing (see [7] for a review of Complex Event Processing). One such early approach is presented in [26]. Complex event patterns are converted to automata, and subsequently, Markov chains are used in order to estimate when a pattern is expected to be fully matched. A similar approach is presented in [2], where again automata and Markov chains are employed in order to provide (future) time intervals during which a match is expected with a probability above a confidence threshold.

Distributed Online Learning. In recent years, the problem of distributed online learning has received increased attention and has been studied in [8, 16, 18, 33, 34]. A distributed online mini-batch prediction approach over multiple data streams has been proposed in [8]. This approach is based on a static synchronization method. The learners periodically communicate their local models with a central coordinator unit after consuming a fixed number of input samples/events (i.e., batch size b), in order to create a global model and share it between all learners. This work has been extended in [16] by introducing a dynamic synchronization scheme that reduces the required communication overhead. It can do so by making the local learners communicate their models only if they diverge from a reference model. In this work, we employ this protocol with event patterns prediction models over multiple event streams.

2.2 Technological Background

In the last years, many systems for large-scale and distributed stream processing have been proposed, including Spark Streaming [12], Apache Storm [14] and Apache Flink [13]. These frameworks can ingest and process real-time data streams, published from different distributed message queuing platforms, such as Apache Kafka [11] or Amazon Kinesis [5].

In the datAcron project, the Flink streaming processing engine has been chosen as a primary platform for supporting the streaming operations, based on an internal comparative evaluation of several streaming platforms. Hence, we used it to implement our system. A predecessor distributed online learning framework has already been implemented in the FERARI project [10] based on Apache Storm.

Apache Flink. Apache Flink is an open source project that provides a large-scale, distributed, and stateful stream processing platform [6]. It is one of the most recent and pioneering Big Data processing frameworks. It provides processing models for both streaming and batch data, where the batch processing model is treated as a special case of the streaming one (i.e., finite stream). Flink’s software stack includes the `DataStream` and `DataSet` APIs for processing infinite and finite data, respectively. These two core APIs are built on top of Flink’s core dataflow engine and provide operations on data streams or sets such as mapping, filtering, grouping, etc.

3 SYSTEM OVERVIEW

3.1 Pattern prediction on a single stream

For our work presented in this paper, we use the approach presented in [2]. For the sake of self-containment, we briefly describe this approach in the following, first assuming that only a single stream is consumed and then adjusting for the case of multiple streams. We follow the terminology of [3, 21, 35] to formalize the problem we tackle.

3.1.1 Problem formulation. We define an input event and a stream of input events as follows:

**Definition 3.1.** Each event is defined as a tuple of attributes $e_i = (id, type, r, a_1, a_2,..., a_n)$, where $type$ is the event type attribute that takes a value from a set of finite event types/symbols $\Sigma$, $r$ represents the time when the event tuple was created, the $a_1, a_2,..., a_n$ are spatial or other contextual features (e.g., speed); these features are varying from one application domain to another. The attribute $id$ is a unique identifier that connects the event tuple to an associated domain object.

**Definition 3.2.** A stream $s = (e_1, e_3,..., e_i,...)$ is a time-ordered sequence of events.

A user-defined pattern $P$ is given in the form of a regular expression (i.e., using operators for sequence, disjunction, and iteration) over $\Sigma$ (i.e., event types) [2]. More formally, a pattern is given through the following grammar:

**Definition 3.3.** $P ::= E | P_1.P_2 | P_1 \lor P_2 | P_1^* \lor P_2^*$, where $E \in \Sigma$ is a constant event type, $.$ stands for sequence, $\lor$ for disjunction and $^*$ for Kleene $-$ $\ast$. The pattern $P := E$ is matched by reading...
an event \( e_i \) iff \( e_i.type = E \). The other cases are matched as in standard automata theory.

The problem at hand may then be stated as follows: given a stream \( s \) of low-level events and a pattern \( P \), the goal is to estimate at each new event arrival the number of future events that we will need to wait for until the pattern is satisfied (and therefore a full match is detected).

3.1.2 Proposed approach. As a first step, event patterns are converted to deterministic finite automata (DFA) through standard conversion algorithms [15]. As an example, see Figure 1a for the DFA of the simple sequential pattern \( P = a; d; c \) and an alphabet \( \Sigma = \{a, b, c, d\} \) (note that the DFA has no dead states since we need to handle streams and not strings). The next step is to derive a Markov chain that will be able to provide a probabilistic description of the DFA’s run-time behavior.

Towards this goal, we use Pattern Markov Chains, as was proposed in [27]. Under the assumption that the input stream is generated by an \( m \)-order Markov source, and after performing a transformation step on the initial DFA to handle the \( m^{|P|} \) order (see [27] for more details). It can be shown that there is a direct mapping of the states of the DFA to states of a Markov chain and the transitions of the DFA to transitions of the Markov chain. The transition probabilities are then conditional probabilities on the event types.

We call such a derived Markov chain a Pattern Markov Chain (PMC) of order \( m \) and denote by \( PMC_P^m \), where \( P \) is the initial pattern and \( m \) the assumed order. As an example, see Figure 1b, which depicts the PMC of order 1 for the generated DFA of Figure 1a.

After constructing a PMC, we can use it to calculate the so-called waiting-time distributions. Given a specific state of the PMC, a waiting-time distribution gives us the probability of reaching a set of absorbing states in \( n \) transition from now (absorbing states are states with self-loops and probability equal to 1.0). By mapping the final states of the initial DFA to absorbing states of the PMC (see again Figure 1). Therefore, we can calculate the probability of reaching a final state, or, in other words, of detecting a full match of the original regular expression in \( n \) events from now.

In order to estimate the final forecasts, another step is required, since our aim is not to provide a single future point with the highest probability but an interval. Predictions are given in the form of intervals, as \( I = (start, end) \). The meaning of such an interval is that the DFA is expected to reach a final state sometime in the future between the \( start \) and \( end \) with probability at least some constant threshold \( \theta_{fc} \) (provided by the user). These intervals are estimated by a single-pass algorithm that scans a waiting-time distribution and finds the smallest (in terms of length) interval that has probability exceeds this threshold \( \theta_{fc} \). For example, Figure 2a shows the waiting-time distributions for the non-final states of the DFA in Figure 1, and the computed prediction intervals are depicted in Figure 2b.

The method described above assumes that we know the (possibly conditional) occurrence probabilities of the various event types appearing in a stream (as would be the case with synthetically generated streams). However, this is not always the case in real-world situations. Therefore, it is crucial for a system implementing this method to have the capability to learn the values of the PMC’s transition matrix. One way to do this is to use some part of the stream to obtain the maximum-likelihood estimators for the transition probabilities [4]. If \( \Pi \) is the transition matrix of a Markov chain with a set of states \( Q, \pi_{ij} \) the transition probability from state \( i \) to state \( j \), \( n_{ij} \) the number of observed transitions from state \( i \) to state \( j \), then the maximum likelihood estimator

![Figure 1: DFA and PMC for \( P = a; d; c \) with \( \Sigma = \{a, b, c, d\} \), and order \( m = 1 \).](image1)

![Figure 2: Example of how prediction intervals are produced. \( P = a; d; c \), \( \Sigma = \{a, b, c, d\} \), \( m = 1 \), \( \theta_{fc} = 0.5 \).](image2)
for $\pi_{i,j}$ is given by:

$$\hat{\pi}_{i,j} = \frac{n_{i,j}}{\sum_{k \in Q} n_{i,k}} = \frac{n_{i,j}}{n_i}$$

Executing this learning step on a single node might require a vast amount of time until we arrive at a sufficiently good model. In this paper, we present a distributed method for learning the transition probability matrix.

### 3.2 Pattern prediction on multiple streams

#### 3.2.1 Problem formulation.
Let $O = \{o_1, \ldots, o_K\}$ be a set of $K$ objects (i.e., moving objects) and $S = \{s_1, \ldots, s_L\}$ be a set of real-time streams of events, where $s_i$ is generated by the object $o_i$. Let $\mathcal{P}$ be a user-defined pattern which we want to apply to every stream $s_i$, i.e., each object will have its own DFA.

The setting that is considered in this work is then described in the following: we have $K$ input event streams $S$ and a system consisting of $K$ distributed predictor nodes $n_1, n_2, \ldots, n_K$, each of which consumes an input event stream $s_i \in S$. The goal is to provide timely predictions and be able to do this at large-scale. Each node $n_i$ handles a single event stream $s_i$ associated with a moving object $o_i \in O$. In addition, it maintains a local prediction model $f_i$ for the user-defined pattern $\mathcal{P}$. The $f_i$ model provides the online prediction about the future full match of the pattern $\mathcal{P}$ in $s_i$ for each new arriving event tuple.

In short, we have multiple running instances of an online prediction algorithm on distributed nodes for multiple input event streams. More specifically, the input to our system consists of massive streams of events that describe trajectories of moving vessels in the context of maritime surveillance, where there is one predictor node for each vessel’s event stream.

#### 3.2.2 The proposed approach.
We designed and developed a scalable and distributed pattern prediction system over a massive input event streams of moving objects. As the base prediction model, we use the PMC forecasting method [2]. Moreover, we propose to enable the information exchange between the distributed predictors/learners of the input event streams, by adapting the distributed online prediction protocol of [16] to synchronize the prediction models, i.e., the transitions probability matrix of the PMC predictors.

Algorithm 1 presents the distributed online prediction protocol by dynamic model synchronization on both the predictor nodes and the coordinator. We refer to the PMC’s transition matrix $\Pi_i$ on predictor node $n_i$ by $f_i$. That is, when a predictor $n_i : i \in [k]$ observes an event $e_j$, it revises its internal model state (i.e., $f_i$) and provides a prediction report. Then it checks the local conditions (batch size $b$ and local model divergence from a reference model $f_j$) to decide whether there is a need to synchronize its local model with the coordinator [or not]. $f_i$ is maintained in the predictor node as a copy of the last computed aggregated model $\bar{f}$ from the previous full synchronization step, which is shared between all local predictors/learners. By monitoring the local condition $\|f_i - f_r\|^2 > \Delta$ on all local predictors, we have a guarantee that if none of the local conditions is violated, the divergence (i.e., variance of local models $\delta(f) = \frac{1}{K} \sum_{i=1}^{K} \|f_i - \bar{f}\|^2$) does not exceed the threshold $\Delta$ [16].

On the other hand, the coordinator receives the prediction models from the predictor nodes that requested for model synchronization (violation). Then it tries to keep incrementally querying other nodes for their local prediction models until reaching out all nodes, or the variance of the aggregated model $\bar{f}$ that is computed from the already received models less or equal than the divergence threshold $\Delta$. Finally, the aggregated model $\bar{f}$ is sent back to the predictor nodes that sent their models after the violation or have been queried by the coordinator.

Algorithm 1: Communication-efficient Distributed Online Learning Protocol

| Predictor node $n_i$: at observing event $e_j$ |
| update the prediction model parameters $f_i$ and provide a prediction service |
| if $j \mod b = 0$ and $\|f_i - f_r\|^2 > \Delta$ then |
| send $f_i$ to the Coordinator (violation) |

| Coordinating: |
| receive local models with violation $B = \{f_i\}_{i=1}^m$ |
| while $|B| \neq k$ and $\|\sum_{f_i \in B} \|f_i - \bar{f}\|^2 > \Delta$ do |
| add other nodes that have not reported violation for their models $B \leftarrow \{f_i : f_i \notin B \text{ and } l \in [k]\}$ |
| receive models from nodes in $B$ |
| compute a new global model $\bar{f}$ |
| send $\bar{f}$ to all the predictors in $B$ and set $f_1 \ldots f_m = \bar{f}$ |
| if $|B| = k$ then |
| set a new reference model $f_r \leftarrow \bar{f}$ |

This protocol was introduced for linear models, and has been extended to handle kernelized online learning models [17]. We also employ this protocol for the pattern prediction model, which is internally based on the PMC $\text{PMC}_p$. This allows the distributed $\text{PMC}^m_p$ predictors for multiple event streams to synchronize their models (i.e., the transition probability matrix of each predictor) within the system in a communication-efficient manner.

We propose a synchronization operation for the parameters of the models ($f_i = \Pi_i : i \in [k]$) of the $k$ distributed PMC predictors. The operation is based on distributed the maximum-likelihood estimation [4] for the transition probabilities of the underlying $\text{PMC}^m_p$ models described by:

$$\hat{\pi}_{i,j} = \frac{\sum_{k \in K} n_{k,i,j}}{\sum_{k \in K} \sum_{l \in L} n_{k,i,l}}$$

Moreover, we measure the divergence of local models from the reference model $\|f_k - f_r\|^2$ by calculating the sum of square difference between the transition probabilities $\Pi_{i}$ and $\Pi_{r}$:

$$\|f_k - f_r\|^2 = \sum_{i,j} (\hat{\pi}_{k,i,j} - \hat{\pi}_{r,i,j})^2$$

In general, our approach relies on enabling the collaborative learning among the distributed predictors. Each predictor node receives a stream of events related to a distinct moving object, and the central coordinator is responsible for synchronizing their prediction models using the synchronization operation. Moreover, the predictors they only need to share the parameters of their models, not the consumed event streams.

We assume that the underlying event streams belong to the same distribution and share the same behavior (e.g., mobility patterns). We claim this assumption is reasonable in many application domains: for instance, in the context of maritime surveillance, vessels travel through standard routes, defined by the International Maritime Organization (IMO). Additionally, vessels have similar mobility patterns in specific areas such as moving.
with low speed and multiple turns near the ports [20, 29]. That allows our system to construct a coherent global prediction model dynamically for all input event streams based on merging their local prediction models.

### 3.3 Distributed architecture

Our system consumes as an input an aggregated stream of events coming from a large number of moving objects, which is continuously collected and fed into the system. It allows users to register a pattern $P$ to be monitored over each event stream of a moving object. The output stream consists of original input events and predictions of full matches of $P$, displayed to the end users. Figure 3 presents the overview of our system architecture and its main components.

![System Architecture](image)

**Figure 3: System Architecture.**

The system is composed of three processing units: (i) pre-processing operators that receive the input event stream and perform filtering and ordering operations, before partitioning the input event stream to multiple event streams based on the associated moving object (ii) predictor nodes (learners), which are responsible for maintaining a prediction model for the input event streams. Each prediction node is configured to handle an event stream from the same moving object, in order to provide online predictions for a predefined pattern $P$. (iii) a coordinator node that communicates through Kafka stream channels with the predictors to realize the distributed online learning protocol. It builds a global prediction model, based on the received local models, and then shares it among the predictors.

Our distributed system consists of multiple pre-processing operators, prediction nodes, and a central coordinator node. These units run concurrently and are arranged as a data processing pipeline, depicted in Figure 3. We leverage Apache Kafka as a messaging platform to ingest the input events streams and to publish the resulting streams. Also, it is used as the communication channel between the predictor nodes and the coordinator. Apache Flink is employed to execute the system’s distributed processing units over the input event streams: the pre-processing operators, the prediction units, and the coordinator node. Our system architecture can be modeled as a logical network of processing nodes, organized in the form of a DAG, inspired by the Flink runtime dataflow programs [6].

### 4 IMPLEMENTATION DETAILS

In this section, we briefly describe in detail the implementation of our system on top of Apache Flink and Apache Kafka frameworks. Each of the three sub-modules, described in Section 3.3, have been implemented as Flink operations over the Kafka events stream.

**Pre-processing and Prediction Operators.** Listing 1 shows how the main workflow of the system is implemented as Flink dataflow program.

The system ingests the input events stream from a Kafka cluster that is mapped to a `DataStream` of events, which is then processed by an `EventTuplesMapper` to create tuples of $(id, event)$, where the `id` is associated to the identifier of the moving object. To handle events coming in out of order in a certain margin, the stream of event tuples is processed by a `TimestampAssigner`, it assigns the timestamps for the input events based on the extracted creation time. Afterwards, an ordered stream of event tuples is generated using a process function `EventSorter`.

```
Listing 1: Flink pipeline for local predictors workflow
```

The ordered stream is then transformed to a `keyedEventsStream` by partitioning it, based on the `ids` values, using a `keyBy` operation. A local `predictor` node in a distributed environment is represented by a `map` function over the `keyedEventsStream`. Each parallel instance of the map operator (predictor) always processes all events of the same moving object (i.e., equivalent `id`), and maintains a bounded prediction model (i.e., $PMC^P$) using the Flink’s `Keyed State` 3. The output streams of the moving objects from the parallel instances of the predictor map functions are sent to a new Kafka stream (i.e., same topic name). They then can be processed by other components like visualization or users notifier.

Moreover, the implementation of the `predictor` map function includes the communication with `coordinator` using Kafka streams. At the beginning of the execution, it sends a registration request to the coordinator. Also at the run-time, it sends its local prediction model as synchronization request, or as a response for a resolution request from the coordinator. These communication messages are published into different Kafka topics as depicted in Table 1.

**Coordinator.** It manages the distributed online learning protocol operations, which is also implemented as Flink program. The coordinator receives messages from the local predictors 3.Keyed State in Flink: https://ci.apache.org/projects/flink/flink-docs-release-1.3/dev/stream/state.html#keyed-state
Table 1: Messages to Kafka topics mapping.

<table>
<thead>
<tr>
<th>Message</th>
<th>Kafka Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>RegisterNode, RequestSync, and ResolutionAnswer</td>
<td>LocalToCoordinatorTopicId</td>
</tr>
<tr>
<td>CoordinatorSync and RequestResolution</td>
<td>CoordinatorToLocalTopicId</td>
</tr>
</tbody>
</table>

through a Kafka Stream of a topic named "LocalToCoordinatorTopicId". It is implemented as a single map function over the messages stream, by setting the parallelism level of the Flink program to "1". Increasing the parallelism will scale up the number of parallel coordinator instances, for example, in order to handle different groupings of the input event streams. The map operator of the coordinator handles three message types from the predictors: (i) RegisterNode that contains a registration request for a new predictor node, (ii) RequestSync to receive a local model after violation, (iii) ResolutionAnswer to receive a resolution response from a local predictor node. In addition, it sends CoordinatorSync messages for all predictors after creating a new global prediction model, or RequestResolution to a ask the local predictors for their prediction models.

5 EMPIRICAL EVALUATION

In this section, we evaluate our proposed system by analyzing the predictive performance and communication complexity using real-world event streams provided by the datAcron project in the context of maritime monitoring. The used event streams describe critical points (i.e., synopses) of moving vessels trajectories, which are derived from raw AIS messages as described in [30]. In particular, for our evaluation experiments we used a data set of synopses that contains 4,684,444 critical points of 5055 vessels sailing in the Atlantic Ocean during the period from 1 October 2015 to 31 March 2016.

We used the synopses data set to generate a simulated stream of event tuples i.e., (id, timestamp, longitude, latitude, annotation, speed, heading), which are processed by the system to attach an extra attribute type that represents the event value, where type ∈ Σ, and Σ = Σ1 = {VerySlow, Slow, Moving, Sailing, Stopping}, which is based on a discretization of the speed values. That is, Σ1 includes a simple derived event types based on the speed value that can be used over streams of raw AIS or critical points. Or Σ = Σ2 = {stopStart, stopEnd, changeInSpeedStart, changeInSpeedEnd, slowMotionStart, slowMotionEnd, gapStart, gapEnd, changeInHeading}, which is derived based on the values of the annotation attribute that encodes the extracted trajectory movement events [30]. Σ2 represents the set of possible mobility changes in the vessel’s trajectory [30], each critical point has at least one event. Where in the case of multiple values, we generate duplicate points each of which corresponding to one event in the same order of Σ2.

In our experiments, we monitor a pattern P1 = Sailing with Σ1 that detects when the vessel is underway (sailing). Likewise, we test a second pattern P2 = changeInHeading; gapStart; gapEnd; changeInHeading with Σ2 that describes a potential illegal fishing activity [2].

Experimental setup. We ran our experiments on single-node standalone Flink cluster deployed on an Ubuntu Server 17.04 with Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz X 8 processors and 32GB RAM. We used Apache Flink v1.3.2 and Apache Kafka v0.10.2.1 for our tests.

Evaluation criteria. Our goal is to evaluate our distributed pattern prediction system, which enables the synchronization of prediction models (i.e., PMC models) on the distributed predictor nodes. Our proposed system can operate in three different modes of protocols/schemes of models synchronization: (i) static scheme based on synchronizing the prediction models periodically every b of input events in each stream, (ii) continuous, full synchronization for each incoming event (hypothetical), (iii) dynamic synchronization protocol based on making the predictors communicate their local prediction models periodically but only under condition that the divergence of the local models from a reference model exceeds a variance threshold Δ (recommended).

We compare our proposed system against the isolated prediction mode, in which models are computed on single streams only, and compare the predictive performance in terms of:

(i) Precision = # of correct predictions / # of total predictions is the fraction of the produced predictions that are correct. For each new event in the stream, the predictor provides a prediction interval where the full match of the pattern might occur. Thus, the predictions are temporarily stored until a full match is detected. At that point, all stored prediction intervals are evaluated by considering those intervals where the full match occurred within as correct.

(ii) Spread = end(1) − start(1) is the width of the prediction interval I, which represents the number of events between the start and the end of I.

Moreover, we study the communication cost by measuring the cumulative communication that captures the number of messages, which are required to perform the distributed online learning modes to synchronize the prediction models. Next, we present the experimental results for the patterns P1 = Sailing with an order of m = 2, and P2 = changeInHeading; gapStart; gapEnd; changeInHeading with first order m = 1. All experiments are performed with setting the batch size to 100 (b = 100), the variance threshold of 2 (Δ = 2), 80% as PMC prediction threshold (θP = 80%), and 200 for the maximum spread.

Experimental results. Figure 4 depicts the average precision scores of predictions models (one prediction model per vessel) of all synchronization modes for the first pattern P1 = Sailing, namely, isolated without synchronization, continuous (full-sync), static, and our recommended approach based on the dynamic synchronization scheme. It can be clearly seen that all methods of distributed learning outperform the isolated prediction models. The hypothetical method of full continuous synchronization has the highest precision rates, while the static and dynamic synchronization schemes have close precision scores. Consequently, dynamic synchronization is not much weaker than the static synchronization, but requires much less communication, as explained below.

Figure 5 provides the amount of the accumulated communication that is required by the three modes of the distributed online learning, while the isolated approach does not require any communication between the predictors. These results are shown for P1. As expected, a larger amount of communication is required for the continuous synchronization comparing to the static and dynamic approaches. Also, it can be seen that we can reduce the communication overhead by applying the dynamic synchronization protocol (a reduction by a factor of 100) comparing to the
static synchronization scheme, even with a small variance threshold $\Delta = 2$. Furthermore, the dynamic protocol is still preserving a close predictive performance to the static one (see Figure 4). Therefore, we will only consider the dynamic synchronization and the isolated approach in the evaluation of the second pattern.

In Figure 4, we also noted that the precision is going down in a first phase and stabilizes then. This seems to be counter-intuitive, as the models should improve when getting more data up to a certain point. For explanation, we have investigated the effect of the distributed synchronization of the prediction models on the average spread value. Figure 6 shows the spread results for all approaches. It can be seen that the spread is higher for the distributed learning based methods comparing to the isolated approach. Furthermore, the average spread is decreasing over time until convergence, as result of confidence increase in the models. This may explain the drop in the precision scores from...
the beginning until reaching the convergence. We will investigate further in the interrelation between precision and spread in future work.

For the second, more complex pattern (P₂), we have found that the precision was worse for a distributed model generated over all vessels than in the model created for each vessel in isolation. This indicates that there is no global model describing the behavior of all models consistently. However, when looking at specific groups of vessels, we achieved an improvement in terms of precision. As initial experiment, we only enable the synchronization of the event streams related to trajectories of moving vessels. This work was supported by EU Horizon 2020 datACon project (grant agreement No 687591).

ACKNOWLEDGMENTS

This work was supported by EU Horizon 2020 datACon project (grant agreement No 687591).

REFERENCES

[2] Elias Alevizos, Alexander Artikis, and George Palouras. 2017. Event Forecasting with Pattern Markov Chains (PMC) [2] as the base prediction model on each event stream, and it applies the protocol for distributed online prediction [16] to exchange information between the prediction models over multiple input event streams. Our proposed system has been implemented using Apache Flink and Apache Kafka. In order to show the usefulness and effectiveness of our approach, we empirically tested it against large real-world event streams related to trajectories of moving vessels. As future work, we will address the open issues emerging from the current findings. Firstly, we will study the interrelation between precision and spread scores by validating the approach over synthetic event streams. Secondly, we will investigate the effect of groupings and more patterns in future work.

6 CONCLUSION

In this paper, we have presented a system that provides a distributed pattern prediction over multiple large-scale event streams of moving objects (vessels). The system uses the event forecasting with Pattern Markov Chain (PMC) [2] as the base prediction model on each event stream, and it applies the protocol for distributed online prediction [16] to exchange information between the prediction models over multiple input event streams. Our proposed system has been implemented using Apache Flink and Apache Kafka. In order to show the usefulness and effectiveness of our approach, we empirically tested it against large real-world event streams related to trajectories of moving vessels. As future work, we will address the open issues emerging from the current findings. Firstly, we will study the interrelation between precision and spread scores by validating the approach over synthetic event streams. Secondly, we will investigate the effect of groupings and more patterns in future work.

As future work, we will address the open issues emerging from the current findings. Firstly, we will study the interrelation between precision and spread scores by validating the approach over synthetic event streams. Secondly, we will investigate the effect of groupings and more patterns in future work.

ACKNOWLEDGMENTS

This work was supported by EU Horizon 2020 datACon project (grant agreement No 687591).

REFERENCES

[2] Elias Alevizos, Alexander Artikis, and George Palouras. 2017. Event Forecasting with Pattern Markov Chains (PMC) [2] as the base prediction model on each event stream, and it applies the protocol for distributed online prediction [16] to exchange information between the prediction models over multiple input event streams. Our proposed system has been implemented using Apache Flink and Apache Kafka. In order to show the usefulness and effectiveness of our approach, we empirically tested it against large real-world event streams related to trajectories of moving vessels. As future work, we will address the open issues emerging from the current findings. Firstly, we will study the interrelation between precision and spread scores by validating the approach over synthetic event streams. Secondly, we will investigate the effect of groupings and more patterns in future work.

6 CONCLUSION

In this paper, we have presented a system that provides a distributed pattern prediction over multiple large-scale event streams of moving objects (vessels). The system uses the event forecasting with Pattern Markov Chain (PMC) [2] as the base prediction model on each event stream, and it applies the protocol for distributed online prediction [16] to exchange information between the prediction models over multiple input event streams. Our proposed system has been implemented using Apache Flink and Apache Kafka. In order to show the usefulness and effectiveness of our approach, we empirically tested it against large real-world event streams related to trajectories of moving vessels. As future work, we will address the open issues emerging from the current findings. Firstly, we will study the interrelation between precision and spread scores by validating the approach over synthetic event streams. Secondly, we will investigate the effect of groupings and more patterns in future work.

ACKNOWLEDGMENTS

This work was supported by EU Horizon 2020 datACon project (grant agreement No 687591).

REFERENCES

[2] Elias Alevizos, Alexander Artikis, and George Palouras. 2017. Event Forecasting with Pattern Markov Chains (PMC) [2] as the base prediction model on each event stream, and it applies the protocol for distributed online prediction [16] to exchange information between the prediction models over multiple input event streams. Our proposed system has been implemented using Apache Flink and Apache Kafka. In order to show the usefulness and effectiveness of our approach, we empirically tested it against large real-world event streams related to trajectories of moving vessels. As future work, we will address the open issues emerging from the current findings. Firstly, we will study the interrelation between precision and spread scores by validating the approach over synthetic event streams. Secondly, we will investigate the effect of groupings and more patterns in future work.

6 CONCLUSION

In this paper, we have presented a system that provides a distributed pattern prediction over multiple large-scale event streams of moving objects (vessels). The system uses the event forecasting with Pattern Markov Chain (PMC) [2] as the base prediction model on each event stream, and it applies the protocol for distributed online prediction [16] to exchange information between the prediction models over multiple input event streams. Our proposed system has been implemented using Apache Flink and Apache Kafka. In order to show the usefulness and effectiveness of our approach, we empirically tested it against large real-world event streams related to trajectories of moving vessels. As future work, we will address the open issues emerging from the current findings. Firstly, we will study the interrelation between precision and spread scores by validating the approach over synthetic event streams. Secondly, we will investigate the effect of groupings and more patterns in future work.

ACKNOWLEDGMENTS

This work was supported by EU Horizon 2020 datACon project (grant agreement No 687591).

REFERENCES

[2] Elias Alevizos, Alexander Artikis, and George Palouras. 2017. Event Forecasting with Pattern Markov Chains (PMC) [2] as the base prediction model on each event stream, and it applies the protocol for distributed online prediction [16] to exchange information between the prediction models over multiple input event streams. Our proposed system has been implemented using Apache Flink and Apache Kafka. In order to show the usefulness and effectiveness of our approach, we empirically tested it against large real-world event streams related to trajectories of moving vessels. As future work, we will address the open issues emerging from the current findings. Firstly, we will study the interrelation between precision and spread scores by validating the approach over synthetic event streams. Secondly, we will investigate the effect of groupings and more patterns in future work.

6 CONCLUSION

In this paper, we have presented a system that provides a distributed pattern prediction over multiple large-scale event streams of moving objects (vessels). The system uses the event forecasting with Pattern Markov Chain (PMC) [2] as the base prediction model on each event stream, and it applies the protocol for distributed online prediction [16] to exchange information between the prediction models over multiple input event streams. Our proposed system has been implemented using Apache Flink and Apache Kafka. In order to show the usefulness and effectiveness of our approach, we empirically tested it against large real-world event streams related to trajectories of moving vessels. As future work, we will address the open issues emerging from the current findings. Firstly, we will study the interrelation between precision and spread scores by validating the approach over synthetic event streams. Secondly, we will investigate the effect of groupings and more patterns in future work.

ACKNOWLEDGMENTS

This work was supported by EU Horizon 2020 datACon project (grant agreement No 687591).

REFERENCES

[2] Elias Alevizos, Alexander Artikis, and George Palouras. 2017. Event Forecasting with Pattern Markov Chains (PMC) [2] as the base prediction model on each event stream, and it applies the protocol for distributed online prediction [16] to exchange information between the prediction models over multiple input event streams. Our proposed system has been implemented using Apache Flink and Apache Kafka. In order to show the usefulness and effectiveness of our approach, we empirically tested it against large real-world event streams related to trajectories of moving vessels. As future work, we will address the open issues emerging from the current findings. Firstly, we will study the interrelation between precision and spread scores by validating the approach over synthetic event streams. Secondly, we will investigate the effect of groupings and more patterns in future work.
Mining Vessel Trajectory Data for Patterns of Search and Rescue

Konstantinos Chatzikokolakis*, Dimitrios Zissis†, Giannis Spiliopoulos* and Konstantinos Tserpes**

* MarineTraffic, London, United Kingdom
Email: {konstantinos.chatzikokolakis, giannis.spiliopoulos} @marinetraffic.com
† Department of Product and Systems Design Engineering, University of the Aegean, Syros, Greece
Email: dzissis@aegean.gr
** Department of Informatics and Telematics, Harokopio University of Athens, Greece
Email: ktserpes@hua.com

Figure 1. Visualization of SAR activity in the Mediterranean Sea during July-September 2015

ABSTRACT

The overall aim of this work is to explore the possibility of automatically detecting Search And Rescue (SAR) activity, even when a distress call has not yet been received. For this, we exploit a large volume of historical Automatic Identification System (AIS) data so as to detect SAR activity from vessel trajectories, in a scalable, data-driven supervised way, with no reliance on external sources of information (e.g. coast guard reports). Specifically, we present our approach which is based on a parallelised, nonparametric statistical method (Random Forests), which has proved capable of achieving prediction accuracy rates higher than 77%.

1 INTRODUCTION

For many years, North Africa has served as the jumping off point for refugees and migrants hoping to cross the Mediterranean Sea to Europe. Since the Syrian war in 2011, there has been a rapid increase in the number of people crossing; a trend which is not expected to stop any time soon. According to the UN Refugee Agency, this year alone, at least 2,030 people have died or gone missing on the voyage, with the greatest number of fatalities occurring along the so-called Central Mediterranean Route, through Libya [23]. Although under maritime law, any vessel in the area of a vessel in distress is obliged to offer assistance, numerous national and international missions have been launched on the EU borders and in the international waters of the Mediterranean, so as to assist in Search and Rescue (SAR) operations, such as Operation Mare Nostrum led by Italy, Operation Triton led by Frontex, NATO Operation Sea Guardian and the EU operation Sophia. Many of these operations were not designed with SAR as a primary mission goal. Due to this numerous Non-Governmental Organisations (NGO) have stepped in and have been performing SAR operations in the area; these include
Migrant Offshore Aid Station (MOAS), Doctors Without Borders, Sea-Watch and others. According to the UNHCR an overall 41% of those rescued have been by the NGOs.

Recently though concerns have been raised about the possible interactions between NGOs and smugglers. A report published by the EU agency Frontex stated that there were “clear indications before departure on the precise direction to be followed in order to reach the NGOs’ boats”[4]. According to this same report, during 2015, and the first months of 2016, smuggling groups instructed migrants to make satellite phone calls to the Maritime Rescue Coordination Centre (MRCC) in Rome so as to initiate targeted rescues on the high seas. During this period, SAR operations were mainly undertaken by Italian law enforcement, EUNAVFOR Med or Frontex vessels with NGO vessels involved in less than 5% of the incidents. From June to October 2016, however, the pattern was reversed. “Satellite phone calls to MRCC Rome decreased sharply (down to 10%) and NGO rescue operations rose significantly to more than 40% of all incidents. Since June 2016, a significant number of boats were intercepted or rescued by NGO vessels without any prior distress call and without official information as to the rescue location” according to Frontex [4].

Maritime Domain Awareness (MDA) is the effective understanding of activities, events and threats in the maritime environment that could impact global safety, security, economic activity or the environment [5]. Whilst in the past, MDA had suffered from a lack of data, current tracking technology has transformed the problem into one of an overabundance of data and information. Currently, huge amounts of structured and unstructured data, tracking vessels during their voyages across the seas, are becoming available, mostly due to the Automatic Identification System (AIS) that vessels of specific categories are required to carry. The AIS is a collaborative, self-reporting system that allows maritime vessels to broadcast their information to nearby vessels and coastal based stations [26]. AIS transceivers allow real time information exchange between vessels and shore based stations through digital radio signals transmitted over dedicated channels in VHF band. The major challenge faced today, is exploiting these vast amounts of data and transform it into actionable information. Discovering patterns emerging within these huge datasets is of great importance so as to provide critical insights into the patterns vessels follow during their voyages at sea.

The main objective of our work is to explore the possibility of leveraging these huge mobility datasets so as to automatically detect vessels performing SAR operations. Towards this direction we adopt a practical data mining and machine learning approach which is capable of overcoming the shortcomings and difficulties presented by AIS data (highly skewed, non-uniform, reception errors etc.) [6]. In sum, this work presents novelties on two fronts:

- **Domain Specific**: The overall aim of this work is to explore if it is possible to automatically detect SAR activity from open data (such as AIS), even when a distress call has not been received. This work has an important social impact, as it can help improve coordination of SAR efforts and understanding of implicated activities (e.g. response time).

- **Algorithmic**: We extract patterns of “rescue-like behavior” from billions of records of spatio-temporal (AIS) data and apply Random Forests, which is a parallelised nonparametric statistical method, evaluated as capable of achieving prediction accuracy rates of more than 77%, even when applied to large volumes of highly skewed geospatial data. To the best of the authors knowledge, no previous work has considered deriving SAR activity from AIS data.

The rest of the paper is organized as follows: Section 2 shortly presents previous work in this domain, while Section 3 describes our approach and Section 4 presents the preliminary results while section 5 concludes this paper by briefly outlining the main contributions of this work and suggesting future improvements.

## 2 RELATED WORK

The rise in the availability of larger quantity and better quality mobility data, has increased the interest of researchers in data driven knowledge discovery. Some of the typical mining tasks in the spatio-temporal context include, frequent pattern discovery, trajectory pattern clustering, trajectory classification, forecasting, and outlier detection. Recent works on pattern discovery are based on online event recognition systems that recognize suspicious and illegal vessel activities of compressed routes (i.e., only critical points of routes are preserved)[17]. Although this solution identifies complex events, it does not classify those to specific vessel operations (e.g. tugging, fishing, search and rescue, etc.). The merits of this work have been extended in where vessels’ moving pattern analysis is performed through an ontology-based system[14]. Trajectory classification, includes constructing a model capable of predicting the class labels of moving objects based on their trajectories and other features [9]. Trajectory classification has been applied in many mobility applications and numerous methods have been proposed throughout the given literature, however less attention has been paid to the maritime domain and classifying a vessel’s type with regards to its trajectory. For example, in [9], authors propose a feature generation framework TraClass for trajectory data from satellite images and trace gas measurements, which generates a hierarchy of features by partitioning trajectories and explores two types of clustering: (1) region-based and (2) trajectory-based. In
this paper, hierarchical region-based and trajectory-based clustering after trajectory partitioning is performed, and a vessel classification rate as high as 84.4% is reported, but unfortunately information on how many vessel types are included in the dataset is not provided [9].

Several studies have proved the value of using AIS data for data driven knowledge discovery in this domain [12, 15, 16]. An interesting trajectory classification case that has caught researchers attention, is that of fishing activity detection; especially for applications such as illegal fishing, where the task can be defined as given a ship trajectory T, predict a label y_i for each data point t_i where y_i ∈ {Fishing, NonFishing} [21]. In [21], authors develop three different models to detect potential fishing behavior according to the type of fishing activity; for trawlers a Hidden Markov Model (HMM) is developed using vessel speed as observation variable; for longliners a pattern recognition approach named Lavielle’s algorithm has been applied; and for purse seiners a multi-layered filtering strategy based on vessel speed and operation time was implemented. Validation against expert-labeled datasets showed average detection accuracies of 83% for trawler and longliner, and 97% for purse seiner. Although these methods were designed for wide applicability, high accuracy results are only achieved by preprocessing AIS data, where wrong detections, noise and faulty out-of-bounds data (e.g. observations on land) are previously removed [21]. The use of AIS data poses a series of data management and data processing challenges linked to the treatment of large volumes of data which may heavily reduce the applicability of the approach. Many traditional data mining approaches assume that the underlying data distribution is uniform and spatially continuous. This is not the case for global AIS data, as it is often to have large geographical coverage gaps, message collisions or erroneous messages especially when processing large areas [18, 25].

In [11] Mazzarella, Vespe, Damalas and Osio focus on discovering and characterising fishing areas by exploiting historical AIS data broadcast by fishing vessels. Specifically, they focused on detecting the behavior of fishing boats that are probably actively fishing. The methodology used for the identification of fishing activity was based on assuming a fishing behaviour highly dependent and characterised by speed. Detecting changes and frequency of speed could help identifying which part of the vessel track can be considered as fishing and which not [13]. Their approach relies on DB-SMoT [20] and DBSCAN [3] but unfortunately it is difficult to evaluate the overall accuracy of their results due to the limited availability of ground truth data.

In [24], authors make use of trajectory kernels in combination with a Support Vector Machines (SVM) to detect fishing activity from AIS data, which was collected in a 50km radius around the Port of Rotterdam. For their classification experiments they use the four most common vessel types: cargo ship, tanker, tug and law-enforcement vessel with the best accuracy score being 76.25%. Jiang, Silver, Hu, De Souza, and Matwin in [8], also make use of AIS data and compare Autoencoders with SVMs and Random Forests. In their work they suggest that autoencoders can perform at least as well as and sometimes better than SVM and Random Forests on classification fishing activities, achieving up to 85% accuracy [8]. However, the nature of the autoencoders is to capture as much information as possible and not as much relevant information as possible and since this work utilised only a small dataset it would be difficult to have only a small part of the input that is relevant to the considered problem. Furthermore, SVMs do not work well with categorical features and often fail to handle larger datasets as they pose significant memory requirements and computational complexity in such cases. Other studies indicate the superiority of Random Forests when used for classification tasks, compared to SVMs and back propagation neural networks [10].

Random Forests, which are based on decision trees combined with aggregation and bootstrap ideas, were first introduced by Breiman in 2001 [2]. They are a powerful nonparametric statistical method allowing to consider in a single and versatile framework regression problems, as well as two-class and multi-class classification problems [19]. Random Forests can deal with large numbers of predictor variables even in the presence of complex interactions, and have been applied successfully in genetics, clinical medicine, and bioinformatics within the past few years. Random Forests have been shown to achieve a high prediction accuracy in such applications and to provide descriptive variable importance measures reflecting the impact of each variable in both main effects and interactions [22]. They are considered capable of good accuracy, relatively robust of outliers and noise, can be pararellised and are thus considered suitable data mining algorithm for big data [1, 2].

3 PROPOSED APPROACH
Our aim is to explore the possibility of automatically detecting SAR activity from open data (such as AIS), even when a distress call has not been received. The task can be formulated as given a set of vessel trajectories T, predict a label y_i for each trajectory t_i where y_i ∈ {SAR, Non-SAR}. A trajectory T is a set of AIS messages monitoring a vessel’s movement from a departure port to a destination port.

3.1 Dataset description and processing requirements
According to International Organisation for Migration, more than 360,000 migrants have arrived to EU by sea in 2016, mainly at Italy, Greece and Spain [7]. With respect to the spatial coverage, our analysis has been focused on a bounding box covering the Central Mediterranean Route,
where most of the refugee fatalities have been observed. Figure 2 below illustrates the bounding box taken into account in conjunction with the refugee fatalities in 2016. It should be noted that our approach relies only on AIS data and the migration fatalities dataset visualised in Figure 2 is used only as a reference to define the bounding box area.

![Figure 2. Spatial coverage in conjunction to migration fatalities for 2016](image)

The considered dataset includes the 6 most relevant to navigation AIS messages out of the 27 AIS message types defined in ITU 1371-4 report [26], which are used in approximately 90% of AIS-based scenarios. More specifically, the dataset includes messages of types 1, 2, 3, 5, 18, and 19 out of which 1, 2, 3, 18 and 19 are position reports, including latitude, longitude, speed-over-ground (SOG), course-over-ground (COG), and other fields related to ship movement, while type 5 messages correspond to static-and voyage information, including the IMO identifier, radio call sign, name, ship dimensions, ship and cargo types.

Each vessel's type can be deduced using the information contained in these messages that the vessel is transmitting. This piece of information, typically referred to as AIS SHIPTYPE, usually consists of two digits, the first one ranging from 1-9 indicates the general category of the subject vessel (e.g., Special Category, Passenger, Cargo, etc.), while the second one provides additional information regarding the vessel's type of cargo in certain vessel categories (e.g., Cargo Ships, Tankers, etc.). The vessel's crew or the accountable officer are responsible for correctly entering information into the AIS transponder and although there are explicit types for SAR vessels, it is frequently the case that vessels participating in SAR operations are not declared as such. Furthermore, only the fact that a vessel’s type is SAR does not necessarily infer that each voyage of the vessel is linked to SAR operations (e.g., such vessel could travel between ports for maintenance purposes). Data volume included in our analysis demands large computational power and a parallel processing approach, due to the fact that traditional analytics fail to handle such volumes of data in a considerable time frame. Consequently, we have deployed our approach in Microsoft Azure which is a distributed computing framework capable to process large amount of data fast. Particularly our system included two Head D12v2 nodes, and six D13v2 Worker nodes summing up to a Spark cluster with 56 cores and 392GB memory (in total). The worker nodes have 8 processing cores and 56GB of memory each and the head nodes have 4 processing cores and 28GB of memory each.

3.2 Data processing and analysis

The dataset used for this study consists of all the voyages of 2016 that intersect with the bounding box shown in Figure 2. More specifically this includes 275.657 (SAR and non-SAR, according to the reported AIS SHIPTYPE) voyages made by 12.291 vessels. These correspond to 54.766.629 AIS observations. After processing the initial data we used an algorithmic approach we have introduced in [6], which determines departure and destination port for each AIS message, thus transforming them into specific voyages. Each voyage includes the vessel’s trajectory as well as its static and voyage information described in the previous subsection. Then, a data curation process was performed, to discard voyages with insignificant amount of positions (e.g. statistically too few to be representative). More specifically, all the voyages that included less than 50 positions were removed (as the geographical area selected covers a distance of over 1500 kilometers, trajectories with only 50 reported positions translate to a sample rate of less than one sample per hour). Such voyages suffer from gaps of communication, which will affect the accuracy and the effectiveness of the proposed. After the curation process the dataset included 114.762 voyages, performed by 10.816 vessels, containing 52.505.718 AIS records. However, the SAR data available in this geographic area for 2016 are more than 100-times less compared to the data of non-SAR voyages. More specifically, the dataset includes 114.377 non-SAR voyages of 10.788 vessels which include a total of 52.429.521 AIS messages while the SAR voyages are 385 made by 28 vessels with 75.797 AIS records. For evaluating the approach, the dataset was split into training and test data; the training set included 70% of the SAR voyages and in order to avoid having imbalanced training data or having imbalanced evaluation metric of the classifier (e.g. true positive rate at some false positive threshold), we subsampled the non-SAR voyages (i.e., randomly selecting a subset) included in the training data. Particularly the training data included 1.544 non-SAR voyages and 261 SAR voyages made by 949 and 26 distinct vessels respectively. The rest of the data (i.e. 30% of the SAR voyages and all the non-SAR voyages not included in the training set) constituted our test data.

For all the records in the dataset we filter the following attributes which will be used in our analysis for distinguishing SAR patterns:

- Ship id: This is a unique identifier for each vessel
b. Ship type: This is a two-digit code that corresponds to the general category of the vessel and the vessel’s type of cargo in certain vessel categories.

c. Latitude, Longitude: These represent the geographic location of the vessel.

d. SOG: This is the speed over ground of the vessel measured in knots.

e. COG: This is the course over ground of the vessel measured in degrees with 0 corresponding to north.

f. Heading: This attribute represents the ship's heading in degrees with 0 corresponding to north.

g. Timestamp: This is the full UTC timestamp that the AIS message was received by MarineTraffic.

It should be noted that COG and Heading may be different, due to weather conditions such as wind speed and direction, wave height and currents (e.g., when vessels are drifting). COG on the one hand is the actual moving direction of the vessel, while heading simply indicates where the ship is pointing compared to north. Based on all these attributes and in conjunction with other datasets that assist on determining the boundaries of a port the following additional attributes were calculated:

a. Departure port id: This is a unique identifier of the port from which the vessel departed.

b. Departure timestamp: Full timestamp of the first AIS message outside of departure port geometry.

c. Departure port name: This is the name of the departure port.

d. Departure port type: This attribute determines the type of the port (e.g., port, anchorage, etc.)

e. Departure country code: This attribute indicates the country of the departure port.

Similar attributes related the arrival of each vessel to a port have been also calculated.

3.3 SAR Motion analysis

All these attributes have been used to transform raw positional data into vessel voyages. However, in order to distinguish SAR trajectories from other voyages it has been required to delve into more details on the motion patterns during SAR operations and focus on maneuverability of such vessels. The methodology used for the identification of SAR activity is based on assuming that SAR behaviour is highly dependent and characterised by frequency of speed changes, frequency of turns, departing and arriving at the same port or anchorage and voyage duration. Detecting changes and frequency of speed as well as departing and arriving at the same port will help distinguishing SAR trajectories from typical voyages (i.e., travelling from one port to another).

However, there are also other types of ships that may follow similar patterns. For instance, inland vessels tend to have frequent changes in course over ground and heading due to the voyage area topography. Another example are tugboats that maneuver other vessels by pushing or towing them. Such vessels typically operate in crowded port or narrow canals and perform various maneuvers leading to increased frequency of turns. Furthermore, tugs typically have the same departure and arrival port as they are called to leave a port (i.e., depart), reach the vessel to be towed (or pushed) and return to the same port. One of the distinguishing factors between such vessels and SAR is the voyage duration. In many SAR operations, once vessels recover migrants from sea, they return to the same port from which they departed so as to disembark rescued people and return back to the SAR operation area. Furthermore, SAR vessels patrolling tend to have a steady course, while when they are engaged in rescuing operation they perform complex maneuvers to collect migrants. In some cases, it has been observed that vessels patrolling an area, may be at open sea (i.e., outside of port boundaries) for several days (or even weeks) traveling in a rather small bounding box (compared to the overall time of their voyage).

Based on those characteristics, we produced some additional attributes that have been considered as possible features for the classification process. For each voyage we have ordered the AIS messages received chronologically and we calculated COG, SOG and Heading deltas for each pair of (chronologically) consecutive messages. Negative values in the COG delta feature indicate moving to the left, while positive values indicate moving to the right. Similarly, negative values in the SOG delta feature indicate speed decrease, while positive values indicate speed increase. Finally, negative values for the Heading delta imply a turn of ship’s heading to the left, while positive values indicate a turn to the right. In our analysis we use the absolute values of COG, SOG and Heading deltas, which capture the magnitude of change of the corresponding attributes. In addition, two extra features have been added to the dataset. The first one is a Boolean value indicating whether the vessel has the same departure and arrival port has been added to the dataset, while the latter one is the voyage duration.

After constructing these last features, we were able to measure the quantiles for the COG, SOG and Heading deltas and it has been observed that SAR operation voyages have different behavior compared to other voyages. More specifically, non-SAR voyages seem to have low values even for large quantiles (i.e., 75%, 80%, 85% etc.) compared to the SAR voyages, meaning that in most observations the COG, SOG and Heading deltas are typically small, while for SAR voyages those quantiles had large values. Thus, we added to our dataset the 50%, 75%, 85% and 95% quantiles for each of those voyages.
4. RESULTS AND DISCUSSION

The focus of this work is on exploiting large volumes of historical AIS data so as to identify SAR operations from trajectories in a scalable data-driven and supervised way. Our approach is based on a parallelised, non-parametric statistical method, the Random Forests. To evaluate the approaches’ performance, we conducted a series of experiments that showcase its effectiveness to unseen real-world data. Firstly, we applied a multiple fold cross-validation procedure and measured the F1 score. This score given by the Equation (1) below is the weighted average of Precision and Recall taking both false positives and false negatives into account. Then, using the best model derived through the cross-validation procedure the algorithm classified the test data.

\[
\text{F1 Score} = 2 \times \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}
\]

4.1 Random Forest training and validation

The training dataset described in subsection 3.2 has been used to train and validate the Random Forest model using the features analysed in subsections 3.2 and 3.3. The dataset has been repeatedly partitioned, following the well-known k-fold cross-validation procedure, into training and validation pairs. The partitioning process has been repeated 5 times (i.e. 5-fold cross validation) each time leading to different training and validation pairs. In each partition we have split the dataset into five parts. Four of them used as training set and one of them as validation set with the former set utilised to create the model of the Random Forest and the latter one used for predicting the class of the observations and comparing it against its actual value. Each Random Forest model derived has 10,000 trees and the F1 metric has been measured, leading to an average score of 0.946 for all the 5 folds. The best model derived from the cross-validation process has been retained and used for predicting the values of the test set. Finally, it should be noted that, although classification has not been applied afore for SAR missions, the Random Forest algorithm shows similar performance compared to other classification schemes used for identifying other types of vessels’ motion patterns such as fishing [8][21][24].

4.2 Random Forest prediction model evaluation

The best model obtained through the 5-fold cross validation process has been used for predicting the labels of the test dataset. To evaluate the performance of the model against first seen data, we measured the F1 score, the Accuracy, the weighted Recall and the weighted Precision presented in Table 1 below. Accuracy is the most intuitive performance measure giving the ratio of correctly predicted observation to the total observations. Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. High Precision relates to the low false positive rate. Finally, Recall is the ratio of correctly predicted positive observations to the all observations in actual class.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1 score</td>
<td>0.986</td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.975</td>
</tr>
<tr>
<td>Weighted Recall</td>
<td>0.975</td>
</tr>
<tr>
<td>Weighted Precision</td>
<td>0.998</td>
</tr>
</tbody>
</table>

The results, show high scores in all the metrics. This occurs due to the highly imbalanced test dataset. More specifically it shows that the model can distinguish non-SAR voyages and classify them as such. The ROC curve and the Area Under ROC curve shown in Figure 3 below indicate also the capabilities of the derived model to classify SAR and non-SAR voyages, as the area under ROC is equal to 0.86.

![ROC curve and Area Under ROC curve of the Random Forest prediction model](image)

Figure 3: ROC curve and Area Under ROC curve of the Random Forest prediction model

However, since the test dataset is imbalanced, and in order to further investigate how well the algorithm identified SAR voyages we have measured the misclassification rate for each vessel type. Particularly the prediction accuracy of each vessel type class has been derived and Table 2 below includes the top-5 (i.e., with most misclassification) vessel types (i.e. false positives) and the misclassification of SAR voyages (i.e. false negatives). The
results show that the classification model labelled accurately 77.5% of the SAR voyages.

Table 2: Top-5 misclassified vessel classes

<table>
<thead>
<tr>
<th>AIS Vessel type</th>
<th>AIS Vessel type name</th>
<th># voyages</th>
<th>Misclassification rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>51</td>
<td>SAR</td>
<td>124</td>
<td>22.5 (false negatives)</td>
</tr>
<tr>
<td>34</td>
<td>Dive Vessels</td>
<td>40</td>
<td>62.5</td>
</tr>
<tr>
<td>53</td>
<td>Port Tender</td>
<td>10</td>
<td>60</td>
</tr>
<tr>
<td>49</td>
<td>High-Speed Craft I</td>
<td>548</td>
<td>57.6</td>
</tr>
<tr>
<td>40</td>
<td>High-Speed Craft II</td>
<td>435</td>
<td>57.01</td>
</tr>
<tr>
<td>30</td>
<td>Fishing</td>
<td>1021</td>
<td>26.75</td>
</tr>
</tbody>
</table>

Though, the misclassification rate of the non-SAR voyages presented above is high, these classes represent a small portion of the overall test dataset, with only a few tens or hundred voyages. On the other hand, the classification algorithm achieved remarkable accuracy rate reaching up to 99.7% in classes with more voyages in the test set. Table 3 below includes the five vessel types with the most voyages in the test set and the misclassification rate for those vessel types.

Table 3: Top 5 vessels with most voyages

<table>
<thead>
<tr>
<th>AIS Vessel type</th>
<th>AIS Vessel type name</th>
<th># voyages</th>
<th>Misclassification rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>Cargo</td>
<td>32,611</td>
<td>0.3</td>
</tr>
<tr>
<td>60</td>
<td>Passenger</td>
<td>17,253</td>
<td>1.64</td>
</tr>
<tr>
<td>71</td>
<td>Cargo – Hazard A</td>
<td>10,308</td>
<td>0.32</td>
</tr>
<tr>
<td>80</td>
<td>Tanker</td>
<td>9,599</td>
<td>1.43</td>
</tr>
<tr>
<td>69</td>
<td>Passenger</td>
<td>9,057</td>
<td>0.695</td>
</tr>
</tbody>
</table>

5. CONCLUSION AND FUTURE WORK
This work focused on the task of automatically detecting SAR vessels from maritime trajectory data. Specifically, we leveraged a large volume of historical AIS data and described our approach which is based on Random Forests, a parallelized nonparametric statistical method, with no reliance on external sources of information (e.g. coast guard reports), so as to detect vessels performing SAR operations in the Mediterranean Sea. The task was formulated as given a set of ship trajectories $T$, predict a label $y_i$ for each trajectory $t_i$ where $y_i \in \{\text{SAR, Non-SAR}\}$. Our proposed approach proved capable of classifying SAR trajectories at an accuracy higher than 77%. To the best of the authors knowledge, no previous work has considered deriving SAR activity from AIS data in a data driven approach. In the future, we will attempt to reformulate the problem towards a point based approach classification, such that given a ship trajectory $T$, predict a label $y_i$ for each data point $t_n$ where $y_i \in \{\text{SAR, NotSAR}\}$. Based on these labeled points, SAR time per area can possibly be calculated on any given scale.

ACKNOWLEDGEMENT
This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 732310 and by Microsoft Research through a Microsoft Azure for Research Award.

REFERENCES
[14] OBDAIR: Ontology-Based Distributed Framework for Accessing, Integrating and Reasoning with Data in Disparate Data Sources (PDF Download Available):


[23] UN Refugee Agency Refugee and migrant flows through Libya on the rise – report.


DiStRDF: Distributed Spatio-temporal RDF Queries on Spark

Panagiotis Nikitopoulos
Department of Digital Systems
University of Piraeus
nikp@unipi.gr

Christos Doulkeridis
Department of Digital Systems
University of Piraeus
cdoulk@unipi.gr

ABSTRACT
The ever-increasing size of data emanating from mobile devices and sensors, dictates the use of distributed systems for storing and querying the data. Typically, these data sources provide some spatio-temporal information, alongside other useful data. The task of interlinking and interchanging this kind of information is challenging in the case of large heterogeneity of data sources. This issue can be addressed by adopting the RDF data model, proposed by W3C. Hence, with respect to an application scenario which analyzes vast amount of spatio-temporal heterogeneous data, the task of efficiently evaluating spatio-temporal queries on RDF is crucial. In this paper, we address the problem of efficiently processing SPARQL spatio-temporal queries in parallel, by proposing the DiStRDF system. We use Spark, a well-known distributed in-memory processing framework, as the underlying processing engine. On top of it, we devise a set of query execution plans which exploit an 1D encoding scheme for improving the performance of our system. Our experimental evaluation demonstrates the efficiency of DiStRDF system.

1 INTRODUCTION
The Resource Description Framework (RDF) is a specification recommended by W3C\(^1\) for modeling and interchanging data over the web. RDF data is represented as a set of triples (subject, property, object) or (s,p,o), also known as statements. SPARQL is a declarative language for querying RDF data sets. It relies on graph pattern matching queries to extract relevant data. A triple pattern \(tp\) is an RDF triple where variables may occur in subject, predicate or object position.

Nowadays, RDF has become a popular model for linking data from heterogeneous data sources. In the era of ever increasing size of RDF repositories (e.g. Google Knowledge Vault), it is imperative to produce efficient and scalable distributed solutions for RDF processing. Challenging issues, such as high availability and fault tolerance, are common in Big Data systems and need to be thoroughly studied. Existing systems that address these issues, typically operate on a set of computing nodes, where data and processing workloads are distributed among them.

The Hadoop ecosystem is a popular solution for addressing Big Data issues. Efficient performance of SPARQL queries over MapReduce is a challenging task, which has attracted the attention of research community [8, 10, 14, 15, 18]. However, in-memory frameworks, such as Spark, typically provide better performance and scalability. Therefore, Spark has arguably become the most popular platform for parallel, in-memory, data processing.

Even though there exist some first approaches for parallel processing of in-memory RDF data [13, 17, 19], these are not designed either for handling spatial nor spatio-temporal data. Essentially, this means that the spatio-temporal processing cannot be integrated in RDF processing, but must be developed as a pre- or post-processing step. However, this “decoupled” approach misses opportunities for pruning unnecessary data early in the query processing pipeline, and inevitably leads to inferior performance.

As an application scenario consider the case of surveillance data from moving objects (e.g., vessels or aircrafts) collected in real time from various data sources (i.e., radars, satellites) in heterogeneous formats, transformed to RDF format, linked with other external data sets and stored in a distributed storage system. Then, this data need to be efficiently queried, in order to retrieve useful information, such as “which vessels were moving in a particular area during the past month?”. Such data analysis tasks that require advanced spatio-temporal queries over RDF data, are common in the case of datAcron project\(^2\).

In this paper, we focus on spatio-temporal SPARQL queries, which apply a user-defined spatio-temporal constraint, alongside other SPARQL operators, on RDF mobility nodes, i.e. RDF nodes that contain spatio-temporal information. Nodes like these, might be parts of an object’s trajectory, or other events of interest. More specifically, given a spatio-temporal query over horizontally partitioned RDF data, our goal is to provide an efficient and scalable distributed processing engine. For simplicity, for the RDF part of the query, we focus on queries expressed as sets of triple patterns, e.g., \(\{tp_1, tp_2, \ldots, tp_n\}\). Although such sets do not cover the entire SPARQL specification, they compose a significant subset that can be used to express several queries, while they constitute one of the most challenging processing tasks for an RDF processing engine.

To support scalable and efficient management and querying of spatio-temporal RDF data, we propose the DiStRDF system (Distributed Spatio-temporal RDF system) which comprises of two main modules: the DiStRDF Storage Layer and the DiStRDF Processing Layer. The Storage Layer performs the necessary management of data on a persistent storage, in order to enable fast data retrieval and high data availability, even in the case of hardware failures. Naturally, the design of the Storage Layer determines to a great extent the efficiency of the corresponding Processing Layer. Hence, the DiStRDF Storage Layer provides several

\(^1\)https://www.w3.org/

\(^2\)http://www.datacron-project.eu/
options for accessing the stored data, to facilitate the selection of the best option at the time of query execution.

RDF data in DiStRDF is actually stored encoded using unique integer identifiers. To produce these identifiers, we exploit an 1D encoding scheme which injects spatio-temporal information to the stored RDF data. This approach has a significant advantage: we can prune nodes based on spatio-temporal criteria, by simply checking their unique identifiers. The DiStRDF Processing Layer exploits this feature to enable efficient distributed spatio-temporal RDF query processing.

In summary, our contributions can be summarized as follows:

- We present the design and implementation of DiStRDF which is a parallel in-memory and scalable spatio-temporal RDF processing engine, based on Spark.
- We propose the DiStRDF Storage Layer which stores encoded RDF triples and a dictionary of mappings between integer identifiers and RDF resources. Moreover, it provides various options for storing and accessing the encoded RDF data.
- We propose the DiStRDF Processing Layer which exploits an 1D encoding scheme for executing efficient spatio-temporal RDF queries, while providing the ability to choose between different query execution plans.
- We implement both Processing and Storage layers and demonstrate the efficiency of the proposed DiStRDF system.

The rest of the paper is organized as follows: Section 2 provides an overview of related work. Section 3 explains the 1D encoding scheme used for storing spatio-temporal data and introduces the DiStRDF Storage Layer. Then, in Section 4 we describe the DiStRDF Processing Layer, and the logical query plans we have implemented. Section 5 presents our experimental study and Section 6 concludes the paper.

2 RELATED WORK

Even though the topic of parallel processing of large-scale RDF data has attracted much attention recently (cf. [1, 9] for related surveys), there is no work on parallel and distributed processing of spatio-temporal RDF data at scale. Approaches for in-memory, distributed processing of RDF data [13, 17, 19] are related to our work, yet they do not cater for the case of spatio-temporal data represented in RDF. In practice, this means that processing of spatio-temporal RDF queries is “decoupled”, leading to filtering the RDF data based on the RDF graph patterns (without taking into account the spatio-temporal constraints), followed by a refinement step that would exclude from the candidate results, those that do not satisfy the spatio-temporal constraints. Unfortunately, this approach incurs higher processing costs, since a large number of candidate results are only pruned at very last stages of query processing.

Scalable processing of big spatial [5, 21, 23, 24] and spatio-temporal [2, 7] data has been studied recently, however these approaches focus only on the spatial (or spatio-temporal) dimension of data, by enabling efficient retrieval based on spatio-temporal constraints. In case of spatio-temporal RDF data, such solutions would have to resolve the required RDF pattern matching after having identified the data that satisfy the spatio-temporal constraints (also called candidate results). Obviously, this approach leads to wasteful processing, since a high number of candidate results are computed in vain, since they will later be pruned by the RDF pattern matching. Clearly, a more efficient solution would resolve both the spatio-temporal part of the query and the graph patterns at the same time, in order to increase the effectiveness of filtering. This is the approach adopted by our work, and it is provided without the use of specialized distributed indexing structures.

Existing works on spatio-temporal RDF data [3, 6, 11] propose RDF storage and processing solutions over centralized stores, therefore they cannot cope with the voluminous nature of big spatio-temporal RDF data that our approach must handle. Finally, the proposed encoding scheme for spatio-temporal RDF data has similarities to the approach adopted in [12] for spatial RDF data. The main difference is that the temporal dimension cannot be treated as yet another dimension, but requires special handling. In turn, this raises challenges relating to producing compact encoded values, an issue that is studied in detail in [22].

3 THE DISTRDF STORAGE LAYER

The storage layer is responsible for distributed storage of RDF data, represented as RDF triples. As typical in RDF storage systems, we employ a dictionary encoding technique [4] using a mapping table, in order to handle triples of integer values. This allows more efficient processing, since it is easier to index, compress, and process integer values, rather than strings.

However, as most of our RDF data have a spatio-temporal nature, we adopt the special-purpose encoding scheme described in [22]. As in any dictionary encoding scheme, an integer value corresponds to an RDF resource uniquely. In our case of RDF resources corresponding to spatio-temporal entities, we generate integer values in an intentional way, so that they provide an approximate position of the entity in space and time. In this way, we can filter RDF triples during scans using spatio-temporal constraints, as is shown in Section 4. More interestingly, this feature comes "for free", without the need to build and maintain special-purpose (distributed) spatio-temporal indexes. Also, our solution is readily applicable to any distributed RDF processing system that utilizes dictionary encoding.

In the following, we first provide a short description of the 1D encoding scheme used for creating the dictionary (Section 3.1), so that the paper is self-contained. Then, we propose our solution for efficient and scalable storage and management of two pieces of data: (a) the dictionary that maps integer values to RDF resources and vice-versa (Section 3.2), (b) a large set of integer-encoded RDF triples (Section 3.3).

3.1 1D Encoding Scheme

Consider a regular spatial grid that partitions the 2D spatial domain into $2^m = (2^{m/2} \times 2^{m/2})$ equi-sized cells. Also, consider a temporal partitioning $T = \{T_0, T_1, \ldots\}$ of the time domain,
where $T_i$ represents a temporal interval. We make no assumptions on specific properties of the partitioning, i.e., the length (or duration) of temporal partitions can vary, apart from the fact that the partitions are disjoint, they cover the entire time domain ($\bigcup T_i = \mathcal{T}$), and that $T_i$ precedes $T_{i+1}$ in the temporal order. Every temporal partition $T_i$ is associated with a 2D spatial grid. The only restriction is that the identical grid structure (i.e., $2^m$ equi-sized cells) is used for all temporal partitions $T_i$.

To encode the spatio-temporal information into an integer (ID), we consider its binary representation consisting of $b$ bits (Figure 1). We set the most-significant bit to 0 for all IDs of spatio-temporal RDF entities, while it is set to 1 for IDs of all other RDF entities. We also keep $m$ bits to represent the different $2^m$ spatial grid cells. Each spatial cell is assigned an $m$-bit identifier using a space-filling curve (Hilbert curve), in order to produce identifiers that respect the spatial locality of cells. Furthermore, we reserve $k$ bits for assigning unique IDs to different entities in the same spatial cell for the same time partition. As such, the maximum number of entities that fit in a spatio-temporal (3D) cell is $2^k$. This part of the ID is auto-incremented, and is encoded in the rightmost bits. Thus, $m + k$ bits are used for representing the identifiers of spatio-temporal entities of a single temporal partition. The remaining $b - (m + k + 1)$ bits are used for encoding the time, thus we can store $2^{b-(m+k+1)}$ temporal partitions in total.

Example 3.1. In Figure 1, we consider the case of $b=16$, $m=4$, and $k=3$, and the depicted identifier is $2^8 + 2^4 + 2^3 = 306$. The spatial cell in which it belongs is $6 \div 0110$, and the spatial grid contains $2^4 = 16$ cells in total. This encoding can accommodate $2^{b-(m+k+1)} = 2^8 = 256$ temporal partitions.

Given an ID of a spatio-temporal entity, the 3D grid cell enclosing the entity can be retrieved. Also, given a 3D cell, a range of IDs can be computed that correspond to any entity belonging to the cell. The proposed encoding ensures that entities with similar spatio-temporal representations are assigned IDs that belong to small ranges, thus preserving data locality. For example, given a time partition $T_i$, all entities $s(\mathcal{T}_i)$ in $T_i$ belong to the interval $[2i \cdot (2^m + k), 2i+1 \cdot (2^m + k)]$, where $2^m$ is the number of spatial cells, and $2^k$ is the maximum number of objects within each spatial cell. Essentially, $2^i$ is used to shift the intervals, thus we can map the different temporal partitions to different 1D intervals of identifiers. In summary, our encoding: (a) allows to retrieve a spatio-temporal approximation given an ID, and (b) achieves to reflect the spatio-temporal locality in the 1D integer domain, by assigning nearby integer values to entities which are close to each other in the spatio-temporal space. Details on the computation of the identifier as well as various strategies for partitioning the temporal domain dynamically are provided in [22].

3.2 Storing the Dictionary

An efficient storage solution needs to be selected for storing the dictionary, by considering the following requirements:

- The data model of the dictionary is a plain key-value model, where a bi-directional mapping is required between strings and integer values.
- Regarding access patterns, the dictionary is used for lookups (random access) based on some key, and we need to support very efficient retrieval of the respective values, in order to avoid delaying the actual query processing.

- The dictionary should be stored in main-memory to enable fast retrieval.
- The size of the dictionary is expected to be large (related works have reported that its size can be comparable to the size of the RDF triples), therefore we need a distributed storage scheme that scales gracefully with increased data size.
- The dictionary should provide high availability, even in the case of hardware failures.

After putting all the above requirements together, we turn our attention to distributed NoSQL key-value stores that satisfy such requirements. In particular, we opt for the popular Redis®, an open-source, in-memory, distributed key-value store, which fits our purposes.

Redis provides a key-value storage and access model that fits our requirements. In order to support efficient bi-directional lookups, we need to maintain two separate Redis databases: the integer to string mapping and vice-versa. Redis keeps all data and indices in main memory, to enable fast data retrieval. Also, it supports data partitioning and replication, to enhance the capacity and availability of the system.

3.3 Storing RDF Triples

Distributed storage of RDF triples has been well-studied recently, due to the ever-increasing number and size of publicly available RDF data sets. The design of our distributed RDF store is generic and supports different options for storing and accessing data, as shown in Figure 2. Our premise is to make available several different options, and provide the ability for the administrator to select the desired query execution plan. Based on the literature, the following aspects guide the design of our RDF triples storage layer:

- **File Layout:** Typical file layouts include row-based storage (e.g., the case of CSV files) and column-based storage (e.g., Parquet®). Both formats have advantages and disadvantages. For instance, it is well-established in the field of databases that columnar format achieves better compression and performs better for queries that retrieve few columns of a table only, while row format is better for queries that retrieve many columns. Our storage layer stores data in both layouts (CSV and Parquet); the desired layout can be picked at the time of query execution.

Figure 2: Design of distributed RDF storage.

- The dictionary should be stored in main-memory to enable fast retrieval.
- The size of the dictionary is expected to be large (related works have reported that its size can be comparable to the size of the RDF triples), therefore we need a distributed storage scheme that scales gracefully with increased data size.
- The dictionary should provide high availability, even in the case of hardware failures.

- The dictionary should be stored in main-memory to enable fast retrieval.
- The size of the dictionary is expected to be large (related works have reported that its size can be comparable to the size of the RDF triples), therefore we need a distributed storage scheme that scales gracefully with increased data size.
- The dictionary should provide high availability, even in the case of hardware failures.

- The dictionary should be stored in main-memory to enable fast retrieval.
- The size of the dictionary is expected to be large (related works have reported that its size can be comparable to the size of the RDF triples), therefore we need a distributed storage scheme that scales gracefully with increased data size.
- The dictionary should provide high availability, even in the case of hardware failures.

- The dictionary should be stored in main-memory to enable fast retrieval.
- The size of the dictionary is expected to be large (related works have reported that its size can be comparable to the size of the RDF triples), therefore we need a distributed storage scheme that scales gracefully with increased data size.
- The dictionary should provide high availability, even in the case of hardware failures.

- The dictionary should be stored in main-memory to enable fast retrieval.
- The size of the dictionary is expected to be large (related works have reported that its size can be comparable to the size of the RDF triples), therefore we need a distributed storage scheme that scales gracefully with increased data size.
- The dictionary should provide high availability, even in the case of hardware failures.

- The dictionary should be stored in main-memory to enable fast retrieval.
- The size of the dictionary is expected to be large (related works have reported that its size can be comparable to the size of the RDF triples), therefore we need a distributed storage scheme that scales gracefully with increased data size.
- The dictionary should provide high availability, even in the case of hardware failures.
Data Organization: Encoded RDF data can be organized into an one triples table, where each row corresponds to a single encoded RDF statement. Property tables is another approach for organizing RDF data, where the row is expanded to include multiple statements. A single row of the table stores a set of property values which share a common subject. The number of the values that are stored together, can be specified at the design time, while the rest of them are stored as simple leftover triples, in an one triples table. Property tables show good performance when a group of properties always exists for a given resource, thereby avoiding the need of costly joins to reassemble this information. Our storage layer currently supports the storage and handling of both one triples table and property tables.

Data Partitioning: The distribution of triples to storage nodes is also important and typically has a major impact on query performance. As SPARQL queries typically involve many joins, a distribution of triples that does not take into account the access patterns based on the query workload is going to hinder data locality. Inevitably, this will result to large data transfers over the network. Thus, a good data partitioning scheme for RDF data is one that processes large parts of the query locally at a node, avoiding the need of exchanging large intermediate results. With respect to the goal of this study, we expect that all queries are going to have a spatio-temporal constraint. Therefore, we consider as a good practice to partition data based on spatio-temporal criteria. To this end, we exploit the spatio-temporal ID used to encode resources, and range-partition triples based on the spatio-temporal information injected in the encoded value of mobility nodes.

Indexing: With respect to indexed access to disk when loading an RDD in memory, we exploit the predicate pushdown mechanism offered by Spark in combination with Parquet storage. Essentially, this mechanism enables selective access to the stored data, by exploiting filters present in the query, in order to restrict access from disk explicitly on those blocks that contain data matching the existing filters.

In order to satisfy the above requirements, we opt to HDFS for storing RDF triples. HDFS is a generic distributed file system which is optimized for storing large sets of data. It can be used as a data source for Spark applications while supporting several file layouts, such as text and Parquet. HDFS supports partitioning by splitting files into blocks of fixed size (usually 128 MBs). File blocks are stored internally into different cluster nodes and can be processed locally when needed. Furthermore, blocks can be replicated among cluster nodes to support high data availability.

4 THE DISTRDF PROCESSING LAYER

The DiStRDF Processing Layer is a SPARQL query engine that supports scalable and efficient batch RDF query processing over vast-sized, spatio-temporal RDF data. To implement the DiStRDF Processing Layer, a parallel in-memory data processing engine is required. For this purpose, we select Apache Spark [25], since it is the most popular data processing engine, with the widest set of contributors, implementing the MapReduce model in main memory, thereby achieving significant performance gains to competitor systems [20], such as Hadoop.

In the following, we explain the basic Spark query operators which can be used for processing queries expressed as sets of triple patterns. Then, we pick a query example to be studied as a typical use case of the proposed DiStRDF system, in order to demonstrate the overall processing approach and its merits in terms of performance gains. We devise a set of logical plans based on this query, which will be examined later in the experimental evaluation.

4.1 Basic Query Operators

The very basic operators needed for querying RDF data include selection, projection and join. Obviously, these operators do not cover the complete SPARQL specification (e.g., grouping, sorting, etc.), however they cover a wide variety of SPARQL queries, and constitute the fundamental and challenging part of a parallel RDF processing engine.

4.1.1 Selection. The Selection operator (σ) takes as input a triple pattern and returns all RDF triples that match the pattern. In the absence of an index, the Selection operator scans the RDF triples to identify matching triples. In the case of an index or sorted access to data, the Selection operator can be implemented more efficiently and avoid the complete scan.

In Apache Spark, a selection of data based on some filtering condition requires in principle a parallel scan of the input data, and loading in an RDD only the records that match the filtering condition. Spark supports predicate pushdown, namely avoiding reading all records and filtering them, but rather reading only the records that match with the filtering condition. Essentially, Spark dictates to the storage system which records are necessary, and lets the storage system filter them without reading in memory. This is a very powerful feature when combined with a storage format, such as Parquet. We exploit predicate pushdown when reading data from disk to memory, in order to achieve better performance.

4.1.2 Projection. The Projection operator (π) takes as input a subset of Subject, Predicate, and Object, and returns only this subset for all RDF triples. In practice, it is useful for keeping only the part of RDF triples necessary for performing a subsequent processing step. As an example, consider selecting only the Subjects of all triples having predicate p1 and object o1. It should be mentioned that projection pushdown is also supported by Parquet.

4.1.3 Join. The Join operator (⋈) takes as input two instances of RDF sets of triples and associates triples from both sets using some common part of the triples (e.g., Subject, Predicate, or Object), which can also be a variable. Join operators usually result to large amounts of data being exchanged over the network, thus optimizing its processing cost is often crucial. Notice that it is quite common to join one set of triples with itself, such as the case of searching graph patterns, i.e., triples that are linked together.

Given the fact that data is distributed, processing a join operator often requires distributed join processing. This is a challenging operation because it usually results in transferring large amounts of data from one node to another, and this cost can dominate the entire execution cost. As a result, optimizing the processing of joins is a critical factor for a distributed RDF processing engine.

The DiStRDF Processing Layer supports all the aforementioned operators at the logical level. These operators may be physically implemented using various algorithms. Typically, query engines support more than one physical implementations on each of the operators available. As such, it is crucial for a query engine to be able to choose effectively at runtime the most efficient physical implementation available in the system. In DiStRDF Processing
Layer the choice of a physical implementation, depends on a static set of rules (rule-based optimization).

The Spark SQL API implements two main physical join operators: Broadcast Hash Join and Sort-merge Join. These algorithms are described in the following, assuming that datasetA and datasetB are joined together, when the size of datasetB is estimated\(^5\) to be smaller than the size of datasetA:

- **Broadcast Hash Join.** This algorithm is typically more efficient for smaller sizes of datasetB. It broadcasts datasetB to all nodes available in the cluster. Then, each node performs a join operation, using the portion of datasetB available locally. The execution steps of this algorithm are described below:
  1. datasetB is collected at a single node of the cluster (also called driver node).
  2. A hashed structure of datasetB is built locally on the driver node.
  3. Hashed datasetB is broadcast to all the nodes.
  4. The broadcast datasetB is joined with local portions of datasetA in parallel, using the hash join algorithm.

- **Sort-merge Join.** This algorithm performs better for larger sizes of datasetB. It can also be used when the actual size of datasetB is unknown and cannot be estimated accurately. It performs a shuffling (i.e. repartitioning) of both datasetA and datasetB on all nodes of the cluster and then joins together the local subsets. Sort-merge Join is a more decentralized algorithm compared to Broadcast Hash Join, at the cost of potentially higher network bandwidth consumption.

  1. datasetA and datasetB are repartitioned (shuffled) using the same partitioner\(^6\) on their respective join keys. Thus, records from both datasets will reside on the same node, if and only if these records share the same join keys.
  2. Each local subset of datasetA is sorted in parallel on all nodes.
  3. The Sort-merge Join algorithm is applied on the subsets of sorted datasetA and datasetB.

4.2 Spatio-temporal RDF Processing

Consider the case of a spatio-temporal query \( StW \), which is defined by a non spatio-temporal SPARQL query \( Q \), and a spatio-temporal constraint \( q \). Moreover, for ease of presentation, let us restrict the SPARQL query \( Q \) to consist of a set of triple patterns, i.e., \( Q = \{ tp_1, tp_2, \ldots, tp_n \} \). In abstract terms, processing an \( StW \) query consists of two parts: (a) processing the triple patterns \( Q \) to find qualifying triples, and (b) processing the spatio-temporal query \( q \) to find matching entities in spatio-temporal terms. The final result is the intersection of these intermediate results.

The gist of our approach is that given the spatio-temporal one-dimensional encoding introduced in Section 3.1, we can transform the spatio-temporal part of the query \( q \) into a range filter for encoded Subjects in triples, i.e., similar to having an additional triple pattern \( tp_{n+1} \) in \( Q \). This approach has the advantage that it essentially replaces the need for spatio-temporal processing (and any specialized index structure that would be required for efficient processing) with a plain additional filtering constraint that needs to be imposed on resulting triples. The caveat is that this approach can produce false positives, i.e., triples that do not actually match the spatio-temporal constraint \( q \). Therefore, the produced result set needs to go through a refinement phase, in order to discard false positives from the final result set.

For a given \( StW \) query, we construct a logical query plan that practically determines one potential way to process the query, by specifying the execution order of the query operators. Obviously, several queries may be executed using various query plans, each leading to different performance. For instance, the \( StW \) query might be executed by either processing the spatio-temporal part of the query first (denoted ST-First), or by processing the RDF part of the query first (denoted RDF-First). Undoubtedly, other query plans can be produced by changing the order of the triple patterns in the RDF part of the query, but we assume a specific order of execution for the triple patterns to ease exposition and examine only these two options for query plans.

DiSTRDF implements both ST-First and RDF-First logical query plans, which are described in detail in the following.

4.3 Logical Query Plans

To exemplify, Figure 3 demonstrates the case of an \( StW \) query, where the left part shows (part of) an RDF graph and the right part depicts a query on that graph that consists of two triple patterns and a spatio-temporal constraint. Essentially the RDF graph is composed of a mobility node (node 1) which has a specific rdf:type property and a set of observation values, such as its speed, its status, its spatial and temporal information, etc. The query’s goal is to retrieve the mobility nodes which are of type

---

\(^5\) An estimator is built in Spark SQL Catalyst optimizer.

\(^6\) A partitioner is a mechanism that determines the location (i.e. node) of each record, on the repartitioning process.
"Node" and have status "SPEED_CHANGE", while satisfying a spatio-temporal box constraint.

The two triple patterns are marked with 1 and 2. Also, we mark the spatio-temporal constraint with 3, as if it were an additional triple pattern, since we have explained that our encoding permits us to deal with spatio-temporal constraints as a triple pattern, namely a Selection operator on triples.

Figure 4 shows the join graph corresponding to the query. Also, notice that the join graph consists of 3 nodes, which correspond to the 3 triple patterns of the query. The edges connect nodes that share a common variable, in this case: ?s. Essentially, the join graph is a representation of triples pattern joins.

This graph can be transformed to a directed acyclic graph (DAG) in different ways, as shown in Figures 4(b) and (c). In this way physical execution plans are produced, which correspond to alternative ways to execute the query. For example, the DAG in Figure 4(b) corresponds to an RDF-First approach, where the RDF part of the query is processed first, pruning unnecessary triples, and followed by the spatio-temporal part of the query. This plan should be selected when the RDF part of the query is processed first, pruning unnecessary triples, and followed by the spatio-temporal part of the query. Again, this plan is preferred when the spatio-temporal constraint is very selective.

4.3.1 RDF-First Logical Plan. Figure 5a depicts the RDF-First logical plan. This query plan aims to minimize the size of \( R_1 \).

First, the RDF data source (set of RDF triples) is filtered by the RDF query predicate, and then by the spatio-temporal information present in the 1D encoding. These filters produce \( R_1 \). Notice the benefit of the proposed encoding scheme: in the absence of this encoding, it would not be possible to apply the second filter, thus \( R_1 \) would contain data filtered only by the RDF constraint. Thus, we can reduce the size of \( R_1 \) at an early stage of processing.

As described earlier, filtering by spatio-temporal ID information, produces false positives, which need to be refined. For this purpose, we need the spatio-temporal exact information of each entity. Therefore, \( R_2 \) is produced by applying the spatial and temporal predicates on the data source. \( R_2 \) contains all the spatio-temporal information contained in the data source.

Then, a join operator is used between \( R_1 \) and \( R_2 \), in order to add the encoded spatio-temporal information to the intermediate result set.

After the join, the refinement phase takes place, which is the same for all logical query plans: spatio-temporal information is decoded in actual spatial and temporal value, and false positives are eliminated to produce the result set. Practically, only the records that satisfy the spatio-temporal query range predicate are kept. Finally, a projection of the final result set is applied to select only the columns that the user has requested. Obviously, the result needs to be decoded prior to being reported to the user.

4.3.2 Improved RDF-First Logical Plan. Figure 5b depicts the improved RDF-First logical plan. This plan aims to minimize both \( R_1 \) and \( R_2 \) at the cost of an extra filtering step. This is achieved by changing the first operator in the plan, which besides filtering based on the RDF constraint, it also keeps the spatial and temporal information. In this way, the filter operator that produces \( R_2 \) is able to avoid accessing the data source, rather it is produced from the in-memory result (RDD) produced by the previous filter operator.

4.3.3 ST-First Logical Plan. Figure 5c depicts the ST-First logical plan. In this plan, the spatio-temporal filtering is applied first. This query plan aims to minimize the size of \( R_2 \).

As already mentioned, the plan performs a spatio-temporal filter first, based on the information encoded in spatio-temporal ID. \( R_1 \) is produced directly by this filter, while \( R_2 \) is produced by applying a filter to \( R_1 \) to keep only the spatio-temporal information.
5 EXPERIMENTAL EVALUATION

In this section, we present the results of our experimental study. Our algorithms are implemented using Scala 2.11 and Apache Spark 2.1. We deployed our code on a proprietary cluster of 10 physical nodes, each having 64GB RAM and a 6-core 1.7GHz processor. All nodes are running Ubuntu 16.04.

5.1 Experimental Setup

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoding scheme</td>
<td>1D encoding, random encoding</td>
</tr>
<tr>
<td>Logical plans</td>
<td>Improved RDF-First, RDF-First, ST-First</td>
</tr>
<tr>
<td>Physical plans</td>
<td>Sort-merge Join, Broadcast Hash Join</td>
</tr>
<tr>
<td>Data organizations</td>
<td>One-triples table, Property table</td>
</tr>
</tbody>
</table>

Table 1: Experimental setup parameters (default values in bold).

Data sets. We used surveillance and static information from the maritime domain, collected in January 2016, covering the Mediterranean Sea and part of the Atlantic Ocean. We used the RDF ontology described in [16]. The size of the data set is 269,357,225 triples, which translates to approximately 6GB in text format. These triples were encoded to integer values using the method described in Section 3.1 to form the encoded triples data set. Data is stored in HDFS using Parquet file format, to enable efficient access for Spark applications and benefit by columnar storage, compression and predicate pushdown. The Parquet input data set is partitioned to 10 HDFS blocks which results to roughly 3GB size of compressed data.

A dictionary containing the mapping between encoded and decoded values was also created and stored in a Redis cluster instance, running on all 10 nodes of the cluster, with no replication enabled.

Configuration. We configured Spark on YARN, where each executor is set to use one virtual core and 2GB of RAM. One node was set to be the driver node, while the others contain the Spark Executors. All experiments conducted, use 10 Spark Executors, with 5 executor cores each. We also used the Jedis 7 library, to communicate with the Redis cluster instances.

Type of query. We focused our experiments on star spatio-temporal queries, i.e., StW queries having a star RDF predicate on a fixed spatio-temporal constraint. All of our experiments were conducted using the same query parameters, producing a result set of 21 triples.

Algorithms. We have implemented the aforementioned logical and physical plans as described in the previous sections for star spatio-temporal queries. More specifically, we experimented with (a) RDF-First, ST-First, Improved RDF-First logical plans, (b) Broadcast Hash Join, Sort-merge Join physical plans and (c) one-triples table, property tables data organizations. Table 1 summarizes the algorithms used during the experimental evaluation process.

Metrics. Our main evaluation metric was the total execution time of each experiment on the Spark cluster. The actual execution time of our algorithms is presented here, omitting any overhead caused by Spark initialization procedures. Each experiment was run 3 times, and the average execution time is depicted in the charts.

Methodology. First, we study the benefits of the 1D encoding scheme, by conducting experiments using our 1D encoding against random encoding, which is typically used by RDF engines. Then, we evaluate each of the logical plans, using a fixed physical plan for the join operator (Sort Merge Join). In the following, we experiment with the two join physical plans, using only the Improved RDF-First logical plan. Finally, we examine the feature of storing RDF data in property tables, against the one-triples table. Our experimental setup is summarized in Table 1, having in bold the default value of each parameter, unless specified otherwise.

5.2 Results

Benefit of 1D Encoding. Figure 6a depicts the execution time comparing our 1D encoding against a random encoding, for all three logical plans. Clearly, by using the spatio-temporal 1D encoding we are able to prune early a set of triples which do not satisfy the query spatio-temporal constraint. This improves performance by at least 10 seconds. It is expected that this gain will increase for larger data sets. This demonstrates the advantage offered by our deliberate encoding scheme. It is also important to note that this early pruning also exploits the predicate pushdown

https://github.com/xetorthio/jedis
feature of Parquet, resulting in smaller I/O cost, since fewer data will be accessed from HDFS.

Comparing the Performance of Logical Plans. Figure 6a demonstrates the performance comparison of logical plans, when using the 1D encoding scheme. ST-First logical plan performs worst, due to the increased size of input to the join operator. Evidently, the RDF predicate is able to prune many triples, resulting to better performance for the RDF-First alternatives. RDF-First achieves to reduce the input size to the join operator, performing better than ST-RDF. However, the Improved RDF-First algorithm combines the benefits of both RDF-First and ST-First, providing to the join operator the smallest input size. These benefits correspond to the execution time needed by Improved RDF-First, which performs better than all other alternatives.

Comparing the Performance of Physical Join Operators. Figure 6b demonstrates the impact on performance by the selection of a physical join operator. The Sort-merge join operator performs better than the Broadcast join operator due to the large size of input data. Notice that we used 5 CPU cores per executor, which results to plenty of data being exchanged locally on the executor’s shared memory. It is also worth noting that Sort-merge Join algorithm, as implemented by Spark SQL API, performs a re-partitioning of the entire data set, to a user configurable number of partitions. This number was set to be equal to the number of executors (10) during the above experiments.

Figure 6c shows that using the property tables data organization results to much better performance, due to not needing a join operation to evaluate the query results.

6 CONCLUSIONS

In this paper, we present the first parallel and scalable in-memory solution to the problem of spatio-temporal RDF query processing. Our proposed DiStRDF system, which comprises of a Processing and a Storage layer, is designed to benefit by the tools and best practices for handling vast sizes of data. Notable features of the proposed solution include the support for various query execution plans, as well as different storage file types and data organizations. Our experiments demonstrate the performance of our system, which is able to efficiently process simple RDF spatio-temporal queries, in a few seconds.

In the future, we plan to extend our system, to cover a larger part of the SPARQL specification. Furthermore, we plan to improve the performance of DiStRDF Processing Layer by implementing more sophisticated execution plans, based on statistics of the data.

ACKNOWLEDGMENTS

This work is supported by the datAcrón project, which has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 687591.

REFERENCES

[17] Alexander Schätzle, Martin Przyjaciel-Zabloki, Thorsten Berberich, and Georg Laosen. 2015. S2X: Graph-Parallel Querying of RDF with GraphX. In Biomedical Data Management and Graph Online Querying – VLDBJ 2015 Workshops, Big-O(N) and DMAF, Wiesbaden, Ht, USA, August 31 – September 4, 2015, Revised Selected Papers. 155–168.
Short-Term Traffic Forecasting: A Dynamic ST-KNN Model Considering Spatial Heterogeneity and Temporal Non-Stationarity

Shifen Cheng, Feng Lu
State Key Lab of Resources and Environment Information System
Institute of Geographic Sciences and Natural Resources Research, Chinese Academy of Sciences
11A, Datun Road, Chaoyang District, Beijing 100101, P. R. China
{chengsf, luf}@lreis.ac.cn

ABSTRACT

Accurate and robust short-term traffic forecasting is a critical issue in intelligent transportation systems and real-time traffic related applications. Existing short-term traffic forecasting approaches are used to adopt global and static model structures and assume the traffic correlations between adjacent road segments within assigned time periods. Due to the inherent characteristics of spatial heterogeneity and temporal non-stationarity of city traffic, it is rather difficult for these approaches to obtain stable and satisfying results. To overcome the problems of static model structures and quantitatively unclear spatiotemporal dependency relationships, this paper proposes a dynamic spatiotemporal k-nearest neighbor model, named D-ST-KNN, for short-term traffic forecasting. It comprehensively considers the spatial heterogeneity and temporal non-stationarity of city traffic with dynamic spatial neighbors, time windows, spatiotemporal weights and other parameters. First, the sizes of spatial neighbors and the lengths of time windows for traffic influence are automatically determined by cross-correlation and autocorrelation functions, respectively. Second, dynamic spatiotemporal weights are introduced into the distance functions to optimize the search mechanism. Then, dynamic spatiotemporal parameters are established to adapt the continuous change in traffic conditions, including the dynamic number of candidate neighbors and dynamic weight allocation parameters. Finally, the D-ST-KNN model is evaluated using two vehicular speed datasets collected on expressways in California, U.S. and city roads in Beijing, China. Four traditional prediction models are compared with the D-ST-KNN model in terms of the forecasting accuracy and the generalization ability. The results demonstrate that the D-ST-KNN model outperforms existing models in all time periods, especially in the morning period and evening peak period. In addition, the generalization ability of the D-ST-KNN model is also proved.

1 INTRODUCTION

Short-term traffic forecasting, which has an important role in intelligent transportation systems, enables traffic managers to formulate reasonable and efficient strategies for alleviating traffic congestion and optimizing traffic assignments. Short-term traffic forecasting also enables the public to achieve accurate vehicular path planning [29][10].

In the past few decades, researchers have proposed several short-term traffic forecasting models that can be divided into two categories: parametric models and nonparametric models. A parametric model uses an explicit parametric function to quantify the relationship between historical traffic data and predicted traffic data. Considering the stochastic and nonlinear characteristics of traffic, constructing a mathematical model with high accuracy for characterizing traffic characteristics in practice is difficult [1]. Nonparametric models, such as data-driven methods, do not require a priori knowledge and explicit expression of mechanism; thus, they are more suitable for short-term traffic forecasting problems [22] [23] [7] [31].

As a typical nonparametric method, the k-nearest neighbors (KNN) model has received considerable attention. Many scholars have successfully applied the traditional KNN model to short-term traffic prediction [2][17][19][4][16][30][28]. Considering that the evolution of traffic is a spatiotemporal interaction process, traffic conditions of road segments are spatially and mutually affected [6]. Therefore, spatiotemporal relationships between multiple road segments in road networks are considered to improve traffic prediction [25][21][20]. Based on the traditional KNN model, [26] realized an enhanced model with the support of spatiotemporal information and argued that it achieves better performance than the model that employs only temporal information. [27] considered upstream and downstream traffic information and proposed a distributed architecture of a spatiotemporal-weighted KNN model for short-term traffic prediction. [3] employed a spatiotemporal state matrix instead of the traditional time series to describe the traffic state while using a Gaussian weight distance to select the nearest neighbor to improve the KNN model. However, the disadvantages of these ST-KNNs are that the spatiotemporal relation cannot be accurately quantified, which is primarily reflected in the modeling process, the size of the spatial dimension m and the length of time window n of the state space cannot be automatically determined, and some values are artificially set. For example, for m=3, three adjacent road segments are selected; for n=2, the historical data of the first two time steps of the current time step are used to construct samples. When the time series problem is transformed into a supervised machine learning problem, the values of m and n determine the number of selected features. Therefore, manually engineered features can easily cause dimensional disaster prevent the guarantee of the prediction accuracy of the model [15].

The prediction model is usually static, thus, it cannot describe the characteristics of the dynamic change in traffic, which are primarily reflected in the following three aspects: 1) existing studies usually assume that the spatial neighbors and time windows are globally fixed, which indicates that once the number of road segments m associated with the predicted road segment and the length of the time window n are determined, they do not change...
in the spatiotemporal range. Considering the dynamic characteristics of an urban road network, traffic flow in the road network is not a static point but is a moving process from one location to another location. The spatial neighbors of the road segment primarily rely on the current traffic conditions. The number of spatial neighbors is very small if traffic congestion exists but is large during flat peak periods [5]. From the perspective of urban road network heterogeneity, the number of relevant road segments for different road segments also differs; thus, sharing parameter $m$ is difficult in the entire spatial range [29]. The selection of a time window based on a time series is used to determine the length of the historical traffic data to match similar traffic patterns. The traffic data in the historical time step and the current time step must be relevant in the selection process [18]. Due to the dynamic and heterogeneous nature of the road network, even the same road segment, a significant difference is observed in the time series of traffic data in different time periods (such as morning and evening peak periods). That causes the selection of the time window to be dynamic [8]. Thus, the spatial neighbors and time windows that dynamically change over time and space are not easily described with globally fixed spatiotemporal state matrices; thus, there is a need for a dynamic spatiotemporal KNN model to adapt to the characteristics of traffic changes. 2) Existing research considers that different historical data for different time periods have different contributions to the prediction of future traffic conditions. When calculating the distance between two state spaces, the weight distance criterion is usually adopted to assign different weights to each component in the state space. The closer the time window is to the predicted time, the larger the allocated weight; the closer the spatial distance is to the predicted road segment, the greater the assigned weight [3]. However, dynamic changes in the spatial neighbor and the time window not only affect the dimension of the space-time matrix but also cause the intensity of the correlation among different positions to dynamically change over time. Therefore, the influence of different components of traffic data is difficult to characterize with global fixed spatiotemporal weight matrix. 3) To determine the value of the number of similar state spaces $K$, researchers usually employ a cross-validation method to select a suitable value, then share in the entire range of space and time [26] [28]. Due to the difference in traffic patterns in the different time periods and space locations, the global fixed value of $K$ cannot adapt to the dynamic and heterogeneous nature of a road network.

The key to short-term traffic forecasting models is the effective use of the potential spatiotemporal dependencies in the traffic data. The existing KNN models usually assume that the traffic change is a static point process and often disregard its important dynamics and heterogeneous characteristics. As a result, the structure of the prediction model is usually globally fixed in time and space, including the globally fixed spatial neighbor, time window, spatiotemporal weights, and spatiotemporal parameters, such as the traditional KNN model and the spatiotemporal KNN model.

In this paper, we propose a dynamic spatiotemporal KNN model (D-ST-KNN) for short-term traffic prediction considering spatial heterogeneity and temporal non-stationarity of city traffic. First, we investigated the autocorrelation of road traffic to determine the time window required for the traffic data. Second, we used the cross-correlation among different road segments to analyze the spatiotemporal dependencies of traffic and build a dynamic spatial neighbor for each road segment. The dynamic spatiotemporal state matrix is obtained by the dynamic spatial neighbor and the dynamic time window instead of the traditional time series or the static spatiotemporal matrix to characterize the state space. Finally, we introduced the dynamic spatiotemporal weight, dynamic spatiotemporal parameters, and Gaussian weight function to improve the KNN model to adapt to the dynamic and heterogeneous characteristics of the traffic.

The remainder of this paper is organized as follows: Section 2 proposes a D-ST-KNN model that considers the spatial heterogeneity and temporal non-stationarity of city road traffic. The construction of the dynamic spatiotemporal state matrix, weights, and other parameters are also introduced in this section. In Section 3, the dynamic characteristics, prediction performance, and computational efficiency of the presented model are comprehensively validated. The experimental results are also discussed. Section 4 concludes the paper and provides an outlook of future work.

2 METHODOLOGY

In this section, we propose a D-ST-KNN model. Our method is divided into five phases: the data bucket partition, state space definition, distance function definition, optimal neighbor selection, and prediction function definition, which corresponds to Sections 2.1-2.5. First, considering the dynamic nature of traffic, the original spatiotemporal data sets are partitioned according to the time periods to form different data buckets. Second, considering the spatial heterogeneity, each segment of a data bucket is separately processed, and the appropriate spatial neighbors and time windows are selected. The spatiotemporal state matrix is constructed to describe the traffic conditions. Then, we introduce the spatiotemporal weight matrix to define the distance function and measure the distance between the current spatiotemporal state matrix and the historical spatiotemporal state matrix to select the $K$ nearest neighbors. Finally, we integrate these neighbors to obtain the predicted value of the target road segment.

2.1 Data bucket

Considering the non-stationarity and periodicity of traffic data, there are significant differences in the traffic characteristics among different time periods, such as the morning peak period, inter-peak period, and evening peak period. In the same period, the traffic data of same road segment has statistical homogeneity and the traffic pattern tends to be stable with periodic changes, such as different days for the morning peak period, which results in the spatial neighbor, the time window, and spatiotemporal parameters that can be shared. Therefore, we divide the original traffic data $\{vol_t^{ij}, j \in [1, N], t \in [t_0, t_c]\}$ into different time periods to describe the homogeneity in same time period and dynamics in different time periods, where $t_0$ and $t_c$ represent the start time step and the current time step of the time series, and $L_j$ denotes the $j$th road segment.

In the study of urban traffic modeling and prediction, to distinguish the difference among the traffic characteristics in different time periods, [24] divided a day into six time periods (period 1: midnight-6:30 am; period 2: 6:30-10:00; period 3: 10:00-13:30; period 4: 13:30-17:00; period 5:17:00-20:30; period 6:20:30-midnight). The text reveals that the partition is statistically acceptable. Based on this analysis and according to the same strategy, the original traffic data are divided into $M$ different time periods ($M=6$) according to the time dimension, which corresponds to different data buckets. Assuming that the entire traffic data set is $BK$, the
data bucket division must be satisfied:

\[
\begin{align*}
\{ & BK = bk_1 \cup bk_2 \cup \ldots \cup bk_M \\
& bk_i = \{vol_L^j \mid 1 \leq j \leq N, \forall t \in [t_{bk_i}^a, \ldots] \}
\end{align*}
\]

where \( i \in [1, M], o \in [1, M], i \neq o, bk_i \) is the \( i \)th bucket (i.e., bucket 1), and \( \text{vol}_L^j \) is the traffic data of road segment \( L_j \) at time step \( t, t \in [t_{bk_i}^a, \ldots] \) indicates that time step \( t \) is within the corresponding time period of the \( i \)th bucket (i.e., [6:00-6:30], [6:30-10:00]). \( L_j \) denotes the \( j \)th road segment (i.e., Link 1), and \( N \) is the total number of road segments. Note that dividing the original traffic data into different buckets at the pre-processing stage does not have any impact on the analyses and conclusions in this study because the same partitioning strategy were used for all the algorithms that are evaluated.

2.2 Dynamic spatiotemporal state matrix

2.2.1 Dynamic spatial neighborhoods. The dynamic spatial neighborhood is used to determine how the traffic conditions of the predicted road segment are affected by the surrounding road segments in different buckets to determine the correlation among road segments. The traditional method usually calculates the correlation coefficients between the time series of the predicted road segments and the time series of other road segments and sets the threshold to select the relevant road segments [3]. Considering that a road network has multiple internal and external factors, such as the influence of traffic lights, the impact of surrounding road segments on predicted road segments has a certain degree of lag. Therefore, the delayed spatiotemporal relationships cannot be exactly expressed by correlation coefficients. The cross-correlation function is a delayed version of the correlation coefficient function, which measures the correlation coefficients of two time series at a specific lag [14]; therefore, it is more suitable for describing the spatiotemporal dependence of traffic.

Assume that \( bk_i \) is the bucket of the predicted road segment \( L_j \) at time step \( t, t \in [t_{bk_i}^a, t_{bk_i}^b] \). Given the surrounding road segments \( L_v \), the time series of the traffic data for two road segments can be expressed as \( U = \{\text{vol}_L^j \mid t \in [t_{bk_i}^a, t_{bk_i}^b]\} \), \( Z = \{\text{vol}_L^j \mid t \in [t_{bk_i}^a, t_{bk_i}^b]\} \), \( j \in [1, N], v \in [1, N] \), and their cross-correlation at lag \( \varphi \) is defined as follows:

\[
\begin{align*}
cc_f^{bk_i}_{z, \varphi} = E \left( (u_t - \mu_u) (z_t + \varphi - \mu_z) \right) \\
\alpha_u = (u_t - \mu_u)^2 \\
\beta_z = (z_t + \varphi - \mu_z)^2 \\
\sigma_u = \sqrt{\alpha_u} \\
\sigma_z = \sqrt{\beta_z}
\end{align*}
\]

where \( \text{cc}_f^{bk_i}_{z, \varphi} \) is the correlation coefficient between time series \( U \) and \( Z \) at lag \( \varphi \) in bucket \( bk_i \), \( \mu_u \) and \( \mu_z \) are the mean values of \( U \) and \( Z \), respectively, and \( \sigma_u \) and \( \sigma_z \) are the standard deviations of \( U \) and \( Z \), respectively.

In this definition, the cross-correlation function can be regarded as a function of lag, and the lag value that makes the cross-correlation function obtain the maximum value is the average delay time of the surrounding segments to the predicted road segment [29]. The formal definition is expressed as

\[
\psi_{bk_i}^{L_v} = \arg \max_{\varphi} \left( cc_f^{bk_i}_{z, \varphi} \right), v \in [1, N]
\]

where \( \psi_{bk_i}^{L_v} \) is the lag value that maximizes cross-correlation of the surrounding road segment \( L_v \) to the predicted road segment in \( bk_i \), and \( \psi_i^{L_v} \) describes the maximum impact time range of the surrounding segments in different buckets on the predicted road segment, which can be employed for efficient selection of spatial neighbors. Consider the predicted road segment \( L_j \) in \( bk_i \) and its predicted time interval \( \delta \). When the surrounding road segments deliver the traffic flow to the predicted road segments within a given time interval, they influence the predicted road segments, and the road segments beyond this time interval are excluded. Its formal definition is expressed as

\[
R_{bk_i}^{L_j} \leftarrow \{L_v | 0 \leq \psi_{bk_i}^{L_v} \leq \Delta_t, v \in [1, N]\}
\]

where \( R_{bk_i}^{L_j} \) is the set of spatial neighbors of the \( j \)th road segment in the \( i \)th bucket.

2.2.2 Dynamic time windows. Considering that the selection of the time window is based on the time series of the predicted road segment, we can select \( n \) historical traffic data that have a correlation with the predicted road segment. The autocorrelation function is usually employed to measure the correlation between the time series and its delayed version; thus, it can be used for the selection of the time window, i.e., the lag in which the prediction error is minimized can be set as the window size. Note that the lag in the autocorrelation function describes the delay effect of the time series, and the lag described in Section 2.2.1 is used to characterize the delay effect between different time series. Given the time series of the \( j \)th road segment \( L_j \) in \( bk_i \), \( U = \{\text{vol}_L^j \mid t \in [t_{bk_i}^a, t_{bk_i}^b]\} \), the autocorrelation function \( \rho_{bk_i}^j (\delta) \) can be defined as follows:

\[
\rho_{bk_i}^j (\delta) = E \left( (u_t - \mu_u)(u_{t-(2\times\delta+\varphi)} - \mu_u) \right) / \sigma_u^2
\]

Using the autocorrelation function to set the time window entails three steps. First, consider the computational limitations, it is necessary to determine the maximum range of lag. Second, within the range, the parameters of the predictive model are fixed, and cross-validation is performed with different lags. This strategy is based on the fact that the value of the traffic data has a significant correlation within the maximum lag range. Finally, the lag that minimizes the prediction error is chosen as the optimal time window.

2.3 Dynamic spatiotemporal weights

Considering the traffic conditions have significant differences at different time intervals, which results in a change in the spatiotemporal weights with time; the historical data of different time and space will influence the future traffic conditions by a different degree. The dimension of the spatiotemporal weight is related to the spatiotemporal state matrix, and the dynamic change in the spatiotemporal matrix causes the dimension of the spatiotemporal weight matrix to change with different time periods. Based on the traditional weight distance function, we introduce a dynamic spatiotemporal weight in the distance function and optimize the weight distance function to adapt the nearest neighbor similarity measure of the dynamic spatiotemporal matrix.

In the temporal dimension, we use the time interval length (i.e., 5 min interval) to characterize the contribution of different time steps. In the spatial dimension, the spatial correlation (such as cross-correlation) is used to characterize the influence of different
spatial distances. The construction method is described as follows: assume that the predicted road segment \( L_j \) at the current time step \( t_c \) is in data bucket \( bk_i \) and the dimension of the spatiotemporal state matrix is \( m_{bk_i} \times n_{bk_i} \), which is determined by the method provided in Section 2.2. Then, the spatiotemporal state matrix of the current time step can be expressed as \( \chi_{t_c}^{L_j} = (m_{bk_i}^{L_j}, n_{bk_i}^{L_j}) \).

The spatiotemporal matrix of the historical time step \( h_i \) can be defined as \( \chi_{h_i}^{L_j} = (m_{bk_i}^{L_j}, n_{bk_i}^{L_j}) \), where \( m_{bk_i}^{L_j} \) is the spatial dimension of the spatiotemporal state matrix of the \( j \)th predicted road segment in the \( ith \) bucket, which is related to the number of elements in the set of spatial neighbors \( R_{bk_i}^{L_j} \). Moreover, \( n_{bk_i}^{L_j} \) is the temporary dimension of the spatiotemporal state matrix of the \( j \)th predicted road segment in the \( ith \) bucket, which is the size of the time window. The time-weighted matrix is defined as \( W_t^{bk_i} \), and the space-weighted matrix is defined as \( W_s^{bk_i} \). The corresponding elements are \( W_t^{bk_i}(ti, tj), ti \in [1, n_{bk_i}], tj \in [1, m_{bk_i}] \) and \( W_s^{bk_i}(si sj), si \in [1, n_{bk_i}], sj \in [1, m_{bk_i}] \), which represent the time weight value and space weight value, respectively, assigned to the \( j \)th predicted road segment in the \( ith \) bucket. The weight distribution is as follows:

\[
W_t^{bk_i}(ti, tj) = \begin{cases} 
\sum_{t_i=t_j}^{t_i}W_t^{bk_i} & \text{if } t_i = t_j \\
0 & \text{otherwise}
\end{cases} 
\]

(6)

\[
W_s^{bk_i}(si sj) = \begin{cases} 
cc_{L_i, L_j}^{si} & \text{if } s_i = s_j \\
0 & \text{otherwise}
\end{cases} 
\]

(7)

In this definition, the temporal and spatial weights are linearly distributed according to the proximity of the current time step and the predicted road segments. \( cc_{L_i, L_j}^{si} \) is the cross-correlation between the time series of the \( i \) spatial neighbor (whose road segment is \( L_{i} \)) and the predicted road segment \( L_j \). The closer the value is to the predicted time, the greater the weight of the allocation; the greater the relation to the space of the predicted road segment, the greater the weight. By introducing spatiotemporal weights into the original spatiotemporal matrix, the spatiotemporal-weighted state matrices of the current time step \( \Gamma_{t_c}^{L_j} \) and the spatiotemporal-weighted state matrices of the historical time step \( \Gamma_{h_i}^{L_j} \) are denoted by the following:

\[
\Gamma_{t_c}^{L_j} = W_t^{bk_i} \times \chi_{t_c}^{L_j} = (m_{bk_i}^{L_j}, n_{bk_i}^{L_j}) \times W_t^{bk_i} 
\]

(8)

\[
\Gamma_{h_i}^{L_j} = W_s^{bk_i} \times \chi_{h_i}^{L_j} = (m_{bk_i}^{L_j}, n_{bk_i}^{L_j}) \times W_s^{bk_i} 
\]

(9)

By calculating the distance \( dB_{bh_i}(t_{t_c}, h_{h_i}) \) between the historical spatiotemporal state matrix and the current spatiotemporal state matrix, candidate neighbors can be selected. The formula is expressed as

\[
d_{bh_i}(t_{t_c}, h_{h_i}) = \sqrt{trac (\Gamma_{t_c}^{L_j} - \Gamma_{h_i}^{L_j})^2} 
\]

where \( trac \) represents the trace of the matrix.

### 2.4 Dynamic spatiotemporal parameters

In the KNN model, the spatiotemporal parameters include the \( K \) values and the parameters introduced during the method construction (such as the prediction generation functions). The reasonableness of the parameters has substantial influence on the prediction accuracy of the model. The \( K \) value is primarily employed to determine the number of candidate neighbors. If the \( K \) value is too small, the model becomes too complex and overfitting is possible. If the \( K \) value is too large, the model is simpler and underfitting is possible. Considering that the selection of the \( K \) value is significantly influenced by the finite sample nature of the problem, the assignment of its values is usually performed by cross-validation to select the \( K \) value that minimizes the model error [27].

The existing methods usually assume that the \( K \) value is globally fixed. When the \( K \) value is determined, it is shared throughout the entire space and time. In contrast to the existing method, the selection of the \( K \) value in the D-ST-KNN model considers the characteristics of dynamic changes of traffic. Instead of setting a global fixed \( K \) value, we can select the optimal \( K \) value for different buckets, i.e., \( K_{bk_i}, bk_i \in BK, i \in [1, M] \).

To verify these assumptions, we use cross-validation to set the range of \( K \) to \([1, 40]\) and test the effect of different \( K \) values on MAPE of the model in different buckets, as shown in Fig. 1.

[Figure 1: Impact of the number of candidate neighbors \( K_{bk_i} \) on the MAPEs of different data buckets.]

As the \( K \) value increases, the prediction error is gradually reduced. When the \( K \) value attains a certain value, the error of the model begins to stabilize. Thus, the optimal \( K \) value for each bucket can be determined (i.e., \( K_{bk_i} = 27, K_{bk_i} = 23 \)). Compared with different buckets, the \( K \) values dynamically vary with different time periods. The global fixed \( K \) value has difficulty describing the dynamic change in traffic. Therefore, the dynamic \( K \) value proposed in this paper is reasonable. The parameters of the D-ST-KNN model also contain the parameters introduced by the predicted generation function (refer to Section 2.5). The calibration method of the parameter is shown in Section 3.2.

#### 2.5 Predictive function

Due to the spatiotemporal state space, the spatiotemporal weight, and the spatiotemporal parameters dynamically change with different buckets; to adapt to this change, the predictive generation function should also dynamically change. This paper transforms the four types of traditional weight distribution methods to enable them to adapt to the dynamics of traffic, including the inverse distance weight [23], rank-based weight [11] [13], and Gaussian weight [3]. Selecting the best prediction function by comparing the performance of different predictive functions (refer to Section 3.2). Note that the weight referred to in this section is expressed as the weight assigned by the candidate neighbor, whereas the weight in Section 2.3 represents the weight matrix of the weights assigned to each element in the spatiotemporal state matrix.
Assuming that $d_{bk_i}$ is the distance between the $k$th candidate neighbor and the predicted road segment in the $i$th bucket obtained by formula (10), \(vol^{L_j}_{t+c+1}\) the predicted value of the predicted road segment $L_j$ at time step $t_c + 1$ is defined as

$$\frac{vol^{L_j}_{t+c+1}}{vol^{L_j}_{t_c+1}} = \sum_{k=1}^{K} a_{bk_i} \frac{vol^{L_j}_{t+c+1}(k) \times \phi^{L_j}_{bk_i}(k)}{\sum_{k=1}^{K} \phi^{L_j}_{bk_i}(k)}$$

where $t_c \in [t_{bk_i}^1, t_{bk_i}^2]$ is used to map the current time step into the corresponding bucket, is used to determine the number of candidate neighbors for the corresponding bucket, $vol^{L_j}_{t+c+1}(k)$ represents the traffic data of the $k$th candidate neighbor, and $\phi^{L_j}_{bk_i}(k)$ and $\phi^{L_j}_{bk_i}(k)$ represent the weight of the $k$th neighbor of the $j$th predicted road segment in the $i$th bucket. The form is defined as follows:

$$\phi^{L_j}_{bk_i}(k) = \begin{cases} \frac{1}{k v_{bk_i}} & \text{if } k v_{bk_i} \geq 1 \\ \frac{1}{k v_{bk_i}} (K v_{bk_i} - r_q + 1)^2 & \text{if } k v_{bk_i} < 1 \\ \frac{1}{4 a_{bk_i}} \exp\left(-\frac{d_{bk_i}^2}{4 a_{bk_i}}\right) & \text{otherwise} \end{cases}$$

Formula (12) corresponds to equal weights, inverse distance weights, the rank-based weight and the Gaussian weight, where $r_q$ represents the order of the $q$th candidate neighbors, and $a_{bk_i}$ is the spatiotemporal parameter whose value is similar to the value of the previously discussed spatiotemporal parameter $K$, which dynamically values with different time periods. The corresponding parameter calibration is shown in Section 3.2.

### 3 EXPERIMENTS

#### 3.1 Data preparation

In this study, two different data sets are used to evaluate the performance of the prediction model. The first data set is PeMS, which is a high-quality data set with open access. PeMS is extensively applied in the field of traffic prediction. The traffic speed data from 59 consecutive locations on the US 101 freeway from PeMS were downloaded for a total of 60 days; the time period is August 15, 2016, to October 14, 2016 and time interval is 5 min (as shown in Table 1). Each detector represents a position; the positional distribution is shown in Fig. 2. The second data set is the floating car trajectory data obtained from the Beijing road network, which is generated from more than 50,000 vehicles equipped with GPS. The frequency of data acquisition is 5 min, and the time period is March 1, 2012, to April 30, 2012 (as shown in Table 1). In this study, a representative region that contains 30 road segments is used for the experiment with the position distribution shown in Fig. 2. In the two data sets, the last ten days are used as the test data to evaluate the accuracy of the model. The remaining days of data are employed as training data to construct the historical database of the predicting model.

In addition, we normalize the original traffic data and use the ratio of the average traffic speed to the maximum speed limit of each road segment to express the traffic conditions of the road segment. The formula expression is as follows:

$$\overline{v_{i,t}} = \frac{v_{i,t}}{f_{i,max}}, \quad i \in [1, N], \quad t \in [t_0, t_c]$$

where $\overline{v_{i,t}}$ is the normalized speed of the $i$th road segment at time step $t$, $v_{i,t}$ is the real average speed data of the road segment, and $f_{i,max}$ is the speed limit for the $i$th road segment.

### Table 1: Description of the experimental data sets

<table>
<thead>
<tr>
<th>Data set</th>
<th>PeMS</th>
<th>Beijing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time span</td>
<td>8/15/2016–10/14/2016</td>
<td>3/1/2012–4/30/2012</td>
</tr>
<tr>
<td>Time interval</td>
<td>5 min</td>
<td>5 min</td>
</tr>
<tr>
<td>Number of links</td>
<td>59</td>
<td>30</td>
</tr>
</tbody>
</table>

#### 3.2 Variable estimation

3.2.1 Determining the optimal distance function. The distance function is used to measure the similarity among the spatiotemporal state matrices to obtain the historical spatiotemporal matrix, which is similar to the spatiotemporal state matrix of the target road segment. Fig. 3 shows the performance differences of the distance function constructed with different weights. The traditional method directly calculates the Euclidean distance between two spatiotemporal state matrices, which treats the elements in the space state matrix equally. The influence of the historical traffic conditions of different time and space distribution in the prediction of future traffic conditions is difficult to describe, which
yields the lowest performance. The distance function constructed by the Gaussian function assigns weights in the time dimension and space dimension; thus, the performance of the prediction model is significantly improved. However, this method requires additional introduction of the time-weighted parameter \(a_t\) and the space-weighted parameter \(a_s\) in the construction process, which makes calibration of its parameters and the global optimal combination of parameters difficult. We adopt a similar strategy that uses the linear time distribution weight in the time dimension and the spatial correlation between the surrounding road segments and the target road segment to assign weights in spatial dimensions. Then, a dynamic spatiotemporal weight assignment method is constructed that does not require any additional parameters. The dynamic weight distribution has the lowest MAPE, RMSE and MAE, which reflects the high efficiency of the method compared to that of the other two weight distribution methods.

**Table 2: Calibration results of the model parameters**

<table>
<thead>
<tr>
<th>Bucket</th>
<th>Parameters</th>
<th>(K_{bk_i})</th>
<th>(a_{bk_i})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bucket 1</td>
<td></td>
<td>27</td>
<td>0.017</td>
</tr>
<tr>
<td>Bucket 2</td>
<td></td>
<td>23</td>
<td>0.015</td>
</tr>
<tr>
<td>Bucket 3</td>
<td></td>
<td>28</td>
<td>0.011</td>
</tr>
<tr>
<td>Bucket 4</td>
<td></td>
<td>18</td>
<td>0.017</td>
</tr>
<tr>
<td>Bucket 5</td>
<td></td>
<td>17</td>
<td>0.014</td>
</tr>
<tr>
<td>Bucket 6</td>
<td></td>
<td>25</td>
<td>0.019</td>
</tr>
</tbody>
</table>

**Figure 3: Comparison of different distance functions**

**Figure 4: Comparison of different weight allocation methods.**

**Figure 5: Impact of the weight parameter \(a_{bk_i}\) on MAPEs for different data buckets.**

3.3.2 Determining the optimal predictive function. Based on the discussion in the previous sections, we transform four types of weight distribution methods, including equal weight, inverse distance weight, rank-based weight, and Gaussian weight, which are used to integrate the candidate neighbors to obtain the final predicted value. In the process of cross-validation, we fix the other parameters of the model, such as \(K_{bk_i}\) and \(a_{bk_i}\), and calculate the influence of different weight distribution methods on the prediction accuracy of the D-ST-KNN model to obtain the average error of the entire test data set for different weight distribution methods. The results are shown in Fig. 4. The MAPE, RMSE and MAE of the Gaussian weight method are lower than the MAPE, RMSE and MAE of the other three weight distribution methods. In the D-ST-KNN model, we employ the Gaussian function as the weight distribution method for candidate neighbors.

3.3.3 Calibrating hyper-parameters. In the D-ST-KNN model, the hyper-parameters primarily include the number of candidate neighbors \(K_{bk_i}\) and the Gaussian weight parameter \(a_{bk_i}\). In the parameter calibration process, to find the best combination of \(K_{bk_i}\) and \(a_{bk_i}\), that enables the prediction model to obtain the minimum MAPE, we set the range of \(K_{bk_i}\) to \([1, 40]\) and the range of \(a_{bk_i}\) to \([0.001, 0.04]\). We apply the cross-validation method to obtain the optimal combination of the parameters for each bucket. The effect of parameter variation on the prediction accuracy of the D-ST-KNN model can be tested by fixing other parameters of the model. For example, we can fix the values of \(a_{bk_i}\) and test the performance of the prediction model changes with \(K_{bk_i}\) (refer to Section 2.4). Because the impact of parameter \(K_{bk_i}\) on the prediction performance was discussed in Section 2.4, this section focuses on the calibration of parameter \(a_{bk_i}\).

Fig. 5 shows the impact of changes in \(a_{bk_i}\) on the performance of the D-ST-KNN model in different buckets. The trend in Fig. 5 reveals the value of \(a_{bk_i}\) has a significant influence on the prediction performance. For the minimum \(a_{bk_i}\), the prediction error of the model attains the maximum \(a_{bk_i}\). As \(a_{bk_i}\) increases, the prediction error gradually decreases and begins to stabilize. We compare the variation of the parameters among the different buckets. For example, in bucket 1, the optimal value of \(a_{bk_i}\) is 0.017, whereas the optimal value of \(a_{bk_i}\) in bucket 2 is 0.015. The value of \(a_{bk_i}\) also changes dynamically over time. Considering that \(K_{bk_i}\) also changes dynamically with time, the parameters of the D-ST-KNN model change with time. The calibration results of the entire model are listed in Table 2, and the values of \(K_{bk_i}\) are shown in Fig. 5. In this analysis, setting the global fixed parameters is unreasonable when constructing the prediction model. We propose the concept of the data bucket, and the prediction model is constructed in different time periods, which causes the model parameters to change with the time period to adapt to the dynamic nature of traffic.

3.3 Accuracy evaluation

3.3.1 Overall results. Based on the variable estimation, we compare our model with several existing traffic prediction models, including the historical average model (HA), Elman neural network (Elman-NN) [9], traditional KNN model (Original-KNN), and spatiotemporal KNN model (ST-KNN). Fig. 6 shows the prediction performance of different models. The HA model, the Elman-NN model, and the Original-KNN model regard the problem of the traffic prediction as a simple time series problem and disregard...
the influence of the spatial factors on the predicted road segment. Therefore, their prediction performance is lower than the prediction performance of the ST-KNN model and the D-ST-KNN model proposed in this paper by comparing the values of MAPE. The ST-KNN model introduces the spatiotemporal state matrix, which improves the prediction performance of the model. However, this matrix ignores the spatial heterogeneity and the temporal non-stationarity of the road network and cannot describe the essential characteristics of the traffic dynamics using a static ST-KNN model (including global fixed spatiotemporal matrix and global fixed parameters). The D-ST-KNN model constructs models for different time periods by introducing the concept of data buckets. Simultaneously, the dynamic space neighbor, dynamic time window, dynamic spatiotemporal weight, and dynamic spatiotemporal parameters are introduced to construct the D-ST-KNN model, which can adequately adapt to the dynamic changes of traffic conditions. The experimental results indicate that the D-ST-KNN model proposed in this paper is superior to other models.

![Figure 6: Accuracy comparison of different models in the Beijing data set.](image)

### 3.4 Generalization ability evaluation

To evaluate the generalization ability of the D-ST-KNN model, we fix all parameters of the model and compare the performance of the different methods with the test data set from PeMS; the experimental results are shown in Fig. 8. The results indicate that the prediction accuracy of the D-ST-KNN model on the PeMS data set is significantly improved compared with that of the Beijing floating car data set. The data quality of the PeMS data set is relatively complete, and the data collection area is the expressway. Compared with the traffic conditions of the urban road network, the traffic mode is relatively simple with minimal changes, which enables the prediction model to easily represent the regular traffic pattern characteristics. However, the D-ST-KNN model maintains the same prediction trend; in all predicted models, its MAPE, RMSE, and MAE are lower than the other models, which exhibit excellent predictive performance and generalization ability.

![Figure 8: Performance comparison of different models in the PeMS data set.](image)

### 4 SUMMARY AND FUTURE WORK

In this paper, we propose a D-ST-KNN model for short-term traffic prediction. The proposed model considers the spatial heterogeneity and temporal non-stationarity of road networks to adapt to the dynamic characteristics of traffic, including dynamic spatial neighbors, time windows, spatiotemporal weights, and spatiotemporal parameters. With cross-correlation and autocorrelation function computation, the automatic selections of spatial neighbors and the time window are realized, which efficiently solve the dimensionality disaster problem encountered in the existing KNN models. The spatiotemporal weights are integrated into a distance function to help identify candidate neighbors. Time variable parameters are also introduced, including the dynamic number of candidate neighbors and dynamic weight allocation parameters, to further adapt to the dynamic and heterogeneous nature of road networks.
Using real traffic data collected from city roads and inter-city expressways, we calculate the number of spatial neighbors and the time window size of each road segments, which reflects the distinct heterogeneity and non-stationarity of urban road traffic. Then, we validate the performance of the proposed D-ST-KNN model with comparisons to HA, Elman-NN, traditional KNN and spatiotemporal KNN models. The experimental results indicate that the D-ST-KNN model has a higher accuracy on short-term traffic prediction than the existing models. In addition, we explore the local performance of different models in different data buckets and find that all models correspond to the degree of traffic congestion, and the D-ST-KNN model performs better than other models in all time periods, especially in the morning period and evening peak period. To summarize, compared with the existing models, the proposed D-ST-KNN model significantly improves the accuracy of short-term traffic prediction. Furthermore, we compare the performance of different models using the actual traffic data collected from PeMS. The D-ST-KNN model also achieves the best performance, which verifies the generalization ability of the proposed model.

In the follow-up study, the following problems need to be investigated to further improve the D-ST-KNN model. The D-ST-KNN model behaves slightly differently in peak and off-peak time periods. Further improvement of the model performance during peak hours will be a constant challenge. Moreover, a multi-threaded approach could be used to improve the efficiency of D-ST-KNN. A parallel P-D-ST-KNN model on an existing parallel computing framework is expected to alleviate the pressure of real-time computation.

5 ACKNOWLEDGMENTS

This research is supported by the Key Research Program of the Chinese Academy of Sciences (Grant No. ZDRW-ZS-2016-6-3) and the State Key Research Development Program of China (Grant No. 2016YFB0502014). Their supports are gratefully acknowledged. And we also thank the anonymous referees for their helpful comments and suggestions.

REFERENCES