Joaquim Filipe
Jorge Bernardino
Christoph Quix (Eds.)

Communications in Computer and Information Science 814

Data Management Technologies and Applications

6th International Conference, DATA 2017
Madrid, Spain, July 24–26, 2017
Revised Selected Papers

Springer

Communications in Computer and Information Science 814

Joaquim Filipe · Jorge Bernardino
Christoph Quix (Eds.)

Data Management Technologies and Applications

6th International Conference, DATA 2017
Madrid, Spain, July 24–26, 2017
Revised Selected Papers

Springer

Editors
Joaquim Filipe
INSTICC
Polytechnic Institute of Setúbal
Setúbal
Portugal

Christoph Quix
RWTH Aachen University
Aachen
Germany

Jorge Bernardino
University of Coimbra
Coimbra
Portugal

ISSN 1865-0929 ISSN 1865-0937 (electronic)
Communications in Computer and Information Science
ISBN 978-3-319-94808-9 ISBN 978-3-319-94809-6 (eBook)
https://doi.org/10.1007/978-3-319-94809-6

Library of Congress Control Number: 2018947371

Preface

The present book includes extended and revised versions of a set of selected papers from the 6th International Conference on Data Science, Technology and Applications (DATA 2017), held in Madrid, Spain, during July 24–26.

We received 66 paper submissions from 28 countries, of which 18% are included in this book. The papers were selected by the event chairs and their selection is based on a number of criteria that include the classifications and comments provided by the Program Committee members, the session chairs' assessment, and also the program chairs' global view of all papers included in the technical program. The authors of selected papers were then invited to submit a revised and extended version of their papers having at least 30% innovative material.

The purpose of the DATA 2017 was to bring together researchers, engineers, and practitioners interested in databases, big data, data mining, data management, data security, and other aspects of information systems and technology involving advanced applications of data.

The papers selected to be included in this book contribute to the understanding of relevant trends of current research on data science, technology, and applications, including four papers addressing the hot topic of business analytics, focusing on data mining, machine learning, and other approaches to understanding the semantics and formulating models that can be used to better solve problems in areas such as natural language understanding and e-Business; another five papers focused on data management and quality, with applications as varied as rail automation, cultural heritage preservation, learning management systems, e-sports, and cloud storage; finally, three papers approached the important aspect of data security and privacy, including graph database versioning, avoiding access credentials theft using server side database credentials, and improving privacy using personalized anonymization of database query results.

We would like to thank all the authors for their contributions and also the reviewers who helped ensure the quality of this publication.

July 2017

Joaquim Filipe
Jorge Bernardino
Christoph Quix

Organization

Conference Chair

Joaquim Filipe	Polytechnic Institute of Setúbal/INSTICC, Portugal

Program Co-chairs

Jorge Bernardino	Polytechnic Institute of Coimbra, ISEC, Portugal
Christoph Quix	Fraunhofer FIT and RWTH Aachen University, Germany

Program Committee

James Abello	Rutgers, The State University of New Jersey, USA
Muhammad Abulaish	South Asian University, India
Hamideh Afsarmanesh	University of Amsterdam, The Netherlands
Abdulrahman Alarifi	King Abdulaziz City for Science and Technology, Saudi Arabia
Mansour Alsaleh	King Abdulaziz City for Science and Technology, Saudi Arabia
Christos Anagnostopoulos	University of Glasgow, UK
Theodoros Anagnostopoulos	Ordnance Survey, UK
Nicolas Anciaux	Inria Paris-Rocquencourt, France
Keijiro Araki	Kyushu University, Japan
Avi Arampatzis	Democritus University of Thrace, Greece
Bernhard Bauer	University of Augsburg, Germany
Fevzi Belli	Izmir Institute of Technology, Turkey
Karim Benouaret	Université Claude Bernard Lyon 1, France
Leopoldo Bertossi	Carleton University, Canada
Gloria Bordogna	CNR, National Research Council, Italy
Francesco Buccafurri	University of Reggio Calabria, Italy
Yixiang Chen	Software Engineering Institute, China
Byron Choi	Hong Kong Baptist University, SAR China
Stefan Conrad	Heinrich Heine University of Düsseldorf, Germany
Agostino Cortesi	Università Ca' Foscari di Venezia, Italy
Gianni Costa	ICAR-CNR, Italy
Theodore Dalamagas	Athena Research and Innovation Center, Greece
Bruno Defude	Institut Mines Telecom, France
Steven Demurjian	University of Connecticut, USA
Martin Drlik	Constantine the Philosopher University in Nitra, Slovak Republic

Fabien Duchateau	Université Claude Bernard Lyon 1/LIRIS, France
John Easton	University of Birmingham, UK
Todd Eavis	Concordia University, Canada
Tapio Elomaa	Tampere University of Technology, Finland
Mohamed Y. Eltabakh	Worcester Polytechnic Institute, USA
Markus Endres	University of Augsburg, Germany
João C. Ferreira	ISCTE, Portugal
Sergio Firmenich	Universidad Nacional de La Plata, Argentina
Francesco Folino	ICAR-CNR, Italy
Hamido Fujita	Iwate Prefectural University, Japan
Javier Fernández García	Vienna University of Economics and Business, Austria
Jérôme Gensel	Université Grenoble Alpes, France
Paola Giannini	University of Piemonte Orientale, Italy
Giorgos Giannopoulos	Athena Research and Innovation Center, Greece
J. Paul Gibson	Mines-Telecom, Telecom SudParis, France
Boris Glavic	Illinois Institute of Technology Chicago, USA
Janis Grabis	Riga Technical University, Latvia
Aziz Guergachi	Ryerson University, ITM, Canada
Mena Habib	Maastricht University, The Netherlands
Raju Halder	Indian Institute of Technology Patna, India
Slimane Hammoudi	ESEO, MODESTE, France
Andreas Henrich	University of Bamberg, Germany
Tsan-Sheng Hsu	Institute of Information Science, Academia Sinica, Taiwan
Ivan Ivanov	SUNY Empire State College, USA
Wang Jianmin	Tsinghua University, China
Christos Kalloniatis	University of the Aegean, Greece
Konstantinos Kalpakis	University of Maryland Baltimore County, USA
Dimitris Karagiannis	University of Vienna, Austria
Pawel Kasprowski	Silesian University of Technology, Poland
Maurice van Keulen	University of Twente, The Netherlands
Masaru Kitsuregawa	The University of Tokyo/National Institute of Informatics, Japan
Kostas Kolomvatsos	University of Thessaly, Greece
Martin Krulis	Charles University, Czech Republic
Vladimir Kurbalija	University of Novi Sad, Serbia
Jean-Charles Lamirel	LORIA, University of Strasbourg, France
Konstantin Läufer	Loyola University Chicago, USA
Dominique Laurent	Cergy-Pontoise University, ENSEA, France
Raimondas Lencevicius	Nuance Communications, USA
Haikun Liu	Huazhong University of Science and Technology, China
Christos Makris	University of Patras, Greece
Yannis Manolopoulos	Aristotle University, Greece
Yuexin Mao	Deloitte Consulting, USA
Miguel A. Martínez-Prieto	University of Valladolid, Spain

Marius Calin Silaghi	Florida Institute of Technology, USA
Alkis Simitsis	HP Labs, USA
Krishnamoorthy Sivakumar	Washington State University, USA
Spiros Skiadopoulos	University of the Peloponnese, Greece
Yeong-Tae Song	Towson University, USA
Sergey Stupnikov	IPI RAN, Russian Federation
Zbigniew Suraj	University of Rzeszow, Poland
George Tambouratzis	Institute for Language and Speech Processing, Greece
Neamat El Tazi	Cairo University, Egypt
Paolo Terenziani	Università del Piemonte Orientale, Italy
Catarci Tiziana	Università degli Studi di Roma La Sapienza, Italy
Christos Tryfonopoulos	University of the Peloponnese, Greece
Michael Vassilakopoulos	University of Thessaly, Greece
Thanasis Vergoulis	Athena Research and Innovation Center, Greece
Karin Verspoor	University of Melbourne, Australia
José Ríos Viqueira	Universidade de Santiago de Compostela, Spain
Gianluigi Viscusi	EPFL Lausanne, Switzerland
Robert Viseur	UMons Polytech, Belgium
Hannes Voigt	TU Dresden, Germany
Florian Wenzel	EXASOL, Germany
Leandro Krug Wives	Universidade Federal do Rio Grande do Sul, Brazil
Yun Xiong	Fudan University, China
Filip Zavoral	Charles University Prague, Czech Republic
Xiaokun Zhang	Athabasca University, Canada
Jiakui Zhao	State Grid Information and Telecommunication Group of China, China

Additional Reviewers

George Alexiou	R.C. Athena, Greece
Iván García Miranda	University of Valladolid, Spain
Maria Sideri	University of the Aegean, Greece
Nikolaos Tantouris	University of Vienna, Austria
Selma Tekir	Izmir Institute of Technology, Turkey
Michael Walch	University of Vienna, Austria
Konstantinos Zagganas	University of the Peloponnese, Greece

Invited Speakers

Torben Bach Pedersen	Aalborg University, Denmark
Francesco Bonchi	ISI Foundation, Italy
Stefano Ceri	Politecnico di Milano, Italy
Andreas Holzinger	Medical University Graz, Austria

Contents

Business Analytics

An Overview of Transfer Learning Focused on Asymmetric Heterogeneous Approaches

Magda Friedjungová(✉) and Marcel Jiřina

Faculty of Information Technology, Czech Technical University in Prague, Prague, Czech Republic
{magda.friedjungova,marcel.jirina}@fit.cvut.cz

Abstract. In practice we often encounter classification tasks. In order to solve these tasks, we need a sufficient amount of quality data for the construction of an accurate classification model. However, in some cases, the collection of quality data poses a demanding challenge in terms of time and finances. For example in the medical area, we encounter lack of data about patients. Transfer learning introduces the idea that a possible solution can be combining data from different domains represented by different feature spaces relating to the same task. We can also transfer knowledge from a different but related task that has been learned already. This overview focuses on the current progress in the novel area of asymmetric heterogeneous transfer learning. We discuss approaches and methods for solving these types of transfer learning tasks. Furthermore, we mention the most used metrics and the possibility of using metric or similarity learning.

Keywords: Asymmetric heterogeneous transfer learning
Different feature space · Domain adaptation · Survey · Data mining
Metric learning

1 Introduction

Classification tasks are part of data mining and machine learning. A classification task lies in answering the question how to suitably classify data using a constructed classification model. This classification model is constructed using a set of data which contains enough records and reaches quality corresponding to the chosen model. Using this data we are able to construct a classification model with high accuracy and subsequently a high probability of it being the correct solution for our classification task. However, in real-world applications we encounter data that is not of high quality - it may contain missing values (individual instances or even whole attributes missing), contain noise or it can be corrupted in some way. In some cases we can even have lack of data and it might not be possible to gather more. Collecting data is often very demanding in terms of time and finances. A satisfying solution can be to make use of data from different but related domains. These different data are represented by different feature spaces which relate to the same domains as the task being solved. Different feature spaces are two spaces, which are represented by different features (also known as attributes). These spaces can originate from other domains. We can divide them into source (usually contains data used for the training of the model) and target feature spaces (contains data used for the testing of the model). These spaces can represent the source and target domains to distinguish where the data comes from. This overview is an extended version of [1] which was presented at the DATA 2017 conference.

As an example, we can imagine object classification as a typical task which could be solved using transfer learning. Our domain in this case is represented by several pictures. Two images of the same object may be of different dimensions of features because of different resolution, illumination or tilt. The feature space can also be represented using text labels. A combination of pictures and text is also possible. We have to solve feature differences while solving the task of object classification. Another example of a feature problem can be cross-lingual document classification. Labeled English documents are widely available, but labeled Russian documents are much harder to obtain. These documents, English and Russian, do not share the same feature representation. Transfer learning can use natural correspondences between feature spaces in order to create an automated learner for Russian documents.

Methods of transfer learning have proven to be very useful in the above cases. Transfer learning enables us to transfer data between domains in such a way that does not require us to have specialised knowledge of either domain. Furthermore, transfer learning increases usability of poor data, which would be unusable on its own. This is thanks to the possibility of combining data from different domains. Transfer of data or their combination would be possible using manual mapping methods. However, such a solution is both time and human resources consuming. Transfer learning saves us time, work, increases the automatisation of the mapping process and enables us to employ data represented by different feature spaces [2].

A general overview of transfer learning is given in [3] and the newest survey was introduced in [4]. Many terms exist for transfer learning, within this work you can also come across a related term - domain adaptation [3]. Domain adaptation is focused on the development of learning algorithms, which can be easily transferred from one domain to another [5]. The main difference to transfer learning is that domain adaptation works for the same categories with different distributions. Transfer learning works across different categories with different distributions. Transfer learning can also be seen as a set of methods which fall into the category of semantic-meaning based methods for cross-domain data fusion. Data fusion consists of techniques for integration of knowledge from various data in a machine learning and data mining task [2]. Transfer learning concerns itself with knowledge fusion rather than schema mapping and data merging, which are more specific to traditional data fusion and data integration being pursued by the database community [2,6]. For better orientation, we bring an overview of basic definitions crucial to understanding transfer learning.

1.1 Definitions

In this subsection we introduce basic definitions of transfer learning terms. Definitions of "domain", "task" and "transfer learning" are provided by [3], where a general overview of transfer learning is given. Helpful definitions can also be found in [7,8].

Definition 1 *(Domain). A domain D is a two-tuple ($\mathcal{X}$, P(X)), where $\mathcal{X}$ is the feature space of D and P(X) is the marginal probability distribution where $X = x_1, ..., x_n \in \mathcal{X}$.*

In our survey we consider one source domain D_S and one target domain D_T. If these domains are different, then the domains have different feature spaces $\mathcal{X}_S$ and $\mathcal{X}_T$ or different marginal probability distributions $P(X_S)$ and $P(X_T)$ or a combination of both.

Definition 2 *(Task). A task T is a two-tuple (Y, f()) for some given domain D. Y is the label space of D and f() is an objective predictive function for D. f() is sometimes written as a conditional probability distribution P(y|x).*

Function $f()$ is not given, but can be learned from the training data. The training data consists of pairs x_i, y_i, where $x_i \in X$ and $y_i \in Y$.

Definition 3 *(Transfer Learning). Given a set of source domains $D_S = D_{s_1}$, ..., D_{s_n} where n > 0, a target domain, D_T, a set of source tasks $T_S = T_{s_1}, \dots T_{s_n}$ where $T_{s_i} \in T_S$ corresponds with $D_{s_i} \in D_S$, and a target task T_T which corresponds to D_T, transfer learning enable us to learn the target predictive function $f_T()$ in D_T with the information gained from D_S and T_S where $D_T \neq D_S$ and $T_T \neq T_S$.*

This definition of transfer learning allows transfer learning from multiple source domains. The condition $D_S \neq D_T$ and $T_S \neq T_T$ also implies that either $\mathcal{X}_S \neq \mathcal{X}_T$ or $P(X_S) \neq P(X_T)$. For example in cross-lingual document classification this means that different features exist between source document set and target document set, because each set is represented by a different language (for example English versus Russian) or the marginal distributions between sets are different (for example each set is focused on a different topic). For transfer learning to be possible another requirement must be fulfiled - there must exist a relationship between the source and target feature spaces, the domains must be related in some way.

In this survey we focus on the diversity of data from different feature spaces of source and target domains. We search for suitable mappings between this data, which will maintain or decrease the error of the predictive or classification model. In practice, the mapping of data is often solved manually, but in some cases this approach poses a combinatorial problem and almost always requires the presence of a domain expert. In the ideal case it would be beneficial to find such an automatic mapping, which would enable us to map the data between source and target feature spaces in both directions. This research area is called heterogeneous transfer learning. Heterogeneous transfer learning is proposed to handle cases where the task is the same, but the source and target feature spaces are different [3].

Definition 4 *(Heterogeneous Transfer Learning). Given a set of source domains $D_S = D_{s_1}, ..., D_{s_n}$ where $n > 0$, a target domain, D_T, a set of source tasks $T_S = T_{s_1}, ... T_{s_n}$ where $T_{s_i} \in T_S$ corresponds to $D_{s_i} \in D_S$, and a target task T_T which corresponds to D_T, transfer learning enable us the learn the target predictive function $f_T()$ in D_T with the information gained from D_S and T_S where $\mathcal{X}_T \bigcap (\mathcal{X}_{s_1} \bigcup ... \mathcal{X}_{s_n}) = \emptyset$.*

Heterogeneous transfer learning can be perceived as a type of transductive learning [3]. Transductive learning assumes that the source and target domains are different. Transductive learning also assumes that we have labeled source data and unlabeled target data. In practice it is sometimes difficult to fulfil this assumption, we often encounter a lack of labeled source data. A low amount of labeled source data can have negative impact on classification model accuracy. This problem can be solved by using heterogeneous transfer learning which is able to work with different combinations of labeled and unlabeled data. We focus on feature-based heterogeneous transfer learning, which also operates with the assumption that feature spaces between domains are different. By difference, we understand different distributions, representations and dimensionality of data. Heterogeneous transfer learning is a relatively new field of research and finds an application in such domains as text classification, image recognition, activity recognition, defect prediction etc. Various types of transfer learning can be seen in Fig. 1 (altered figure from [1]).

As far as we know, a lot of work has been done on transfer learning in the machine learning field [3,4]. During our research we determined that asymmetric heterogeneous transfer learning methods are a very interesting, yet seldom explored topic. In this survey article we bring an overview of the basic methods used to unite different feature spaces using asymmetric heterogeneous transfer learning. In this survey we do not give an experimental comparison of individual methods and algorithms. Unfortunately, most approaches are domain or task specific, they reach the best results only on specific datasets. Thus it makes it impossible to carry out a quality comparative analysis. We have also encountered the problem that some methods are not publicly accessible (as open-source repositories). However, we hope to provide a useful survey for the machine learning and data mining communities.

The survey is organized as follows. In Sect. 2 we bring an explanation of the heterogeneous transfer learning problem along with prerequisites for the problem of transfer learning between different feature spaces. Section 2.1 briefly describes solutions based on transformation of features to a common feature space. In

Sect. 2.1 we introduce an asymmetric approach. We designate a separate Sect. 3 for the description of individual asymmetric methods, which contains subsections describing methods applied in specific fields. In Sect. 4 we introduce the reader to the most used feature mappings in the field of transfer learning. In the last Sect. 5 we bring a brief summary of this survey and in Sect. 5.1 we discuss possible challenges, which we would like to address in the future.

2 Heterogeneous Transfer Learning

Transfer learning can be categorized into two settings: homogeneous and heterogeneous (Fig. 1). They both depend on the task being solved and feature spaces. The homogeneous transfer learning setting means that the source and target task differ, but the source and target feature input spaces are the same. The heterogeneous transfer learning setting assumes that the source and target task are the same, while the source and target feature spaces are different. We are interested in the heterogeneous transfer learning setting. Heterogeneous transfer learning includes two situations:

1. The feature spaces between the source and target domains are different, $\mathcal{X}_S \neq \mathcal{X}_T$.
2. The feature spaces between the source and target domains are the same, $\mathcal{X}_S = \mathcal{X}_T$, but the marginal probability distributions of the input data are different, $P(X_S) \neq P(X_T)$.

Heterogeneous transfer learning can be divided into two categories according to what we transfer. The first category is transfer learning through instances, also known as instance-based. Instance-based transfer assumes that a certain part of the data in the source domain can be reused for learning in the target domain. The main methods are instance re-weighting and importance sampling. The second category is transfer learning based on features. The main idea is to learn or find a satisfactory feature representation for the target domain [3]. In the remaining part of this paper we will concern ourselves with feature-based methods, because, as we know, they are more novel and less researched. In this paper there is a whole section dedicated to feature mapping methods (Sect. 4).

In Fig. 2 (altered figure from [1,4]) we can see two ways of transforming data on a feature-based level, which are addressed by different feature spaces. By transformation we understand operations (e.g. translation, scaling, etc.) which have to be done to map different feature spaces. One of these is symmetric transformation (Fig. 2a). Symmetric transformation transforms the source and target feature spaces into a common latent feature space in order to unify the input spaces of the domains. The second approach is asymmetric transformation (Fig. 2b), which transforms the source feature space to the target feature space [3]. Some methods presented in Sect. 3 perform the transformation in the opposite direction, from the target domain to the source domain. Some proposed methods are usable in both directions. All presented approaches in Sect. 3 are based on features.

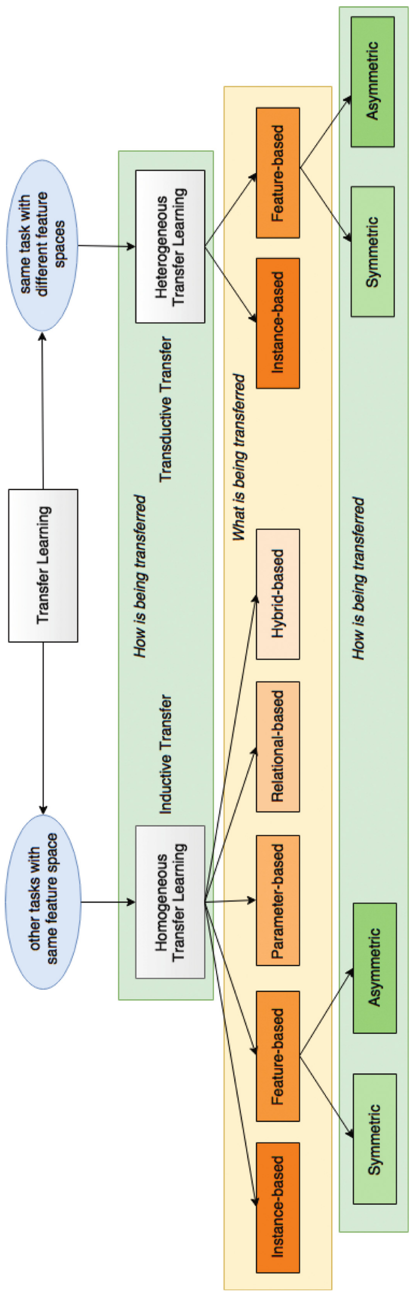


Fig. 1. A hierarchical overview of various transfer learning approaches.

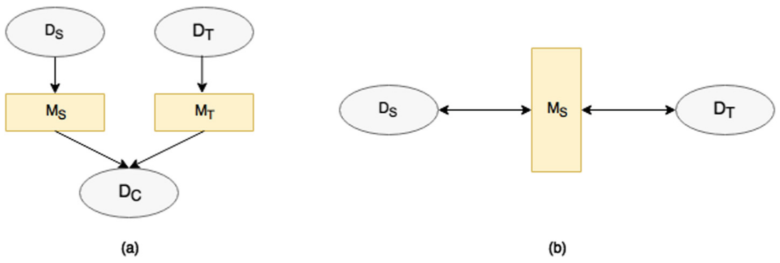


Fig. 2. (a) Symmetric transformation mapping M_S and M_T of the source D_S and target D_T domains into a common latent feature space. (b) Asymmetric transformation mapping M_S of the source domain D_S to the target domain D_T or vice versa from the target D_T to the source D_S domain.

We consider two datasets as a running example in this paper (shown in Fig. 2). The data in these datasets originate from different source and target domains represented by different feature spaces. The datasets can consist of different feature representation, distribution, scale, and density of data. One of the assumptions is that the source and target domains must be related in some way. Thanks to this there exist some relations between source and target feature spaces. These relations can be called correspondences among features. The discovery of as much common information as possible from these different datasets is one of the problems in data mining research. The following sections concern themselves with the individual approaches in more detail.

2.1 Symmetric Feature-Based Approach

Most existing heterogeneous transfer learning methods assume that both source and target data can be represented by a common feature space, called latent feature space. Thus, we are looking for transformation mappings M_S and M_T to transform source domain data D_S and target domain data D_T to a new common latent feature space D_C as is shown in Fig. 2 [9–12]. The whole process of the usage of data in a classification model is described in Fig. 3. Firstly, we have to find a transformation to a common latent feature space D_C for both source D_S and target D_T domain data. We are then able to train and test a model for this D_C space.

There exist a lot of tasks in the natural language processing area. [13] introduce the structural correspondence learning (SCL) algorithm, which extends [14], to automatically induce correspondences among features from different domains. SCL is based on the correlation between certain pivot features. Pivot features are features which behave in the same way for discriminative learning (e.g. classification) in both domains. These features have the same semantic meaning in both domains and are used to learn a mapping from the different feature spaces to a common latent feature space. In [15] they learn a common latent feature space with low dimensionality. The space is learned using a new dimensionality

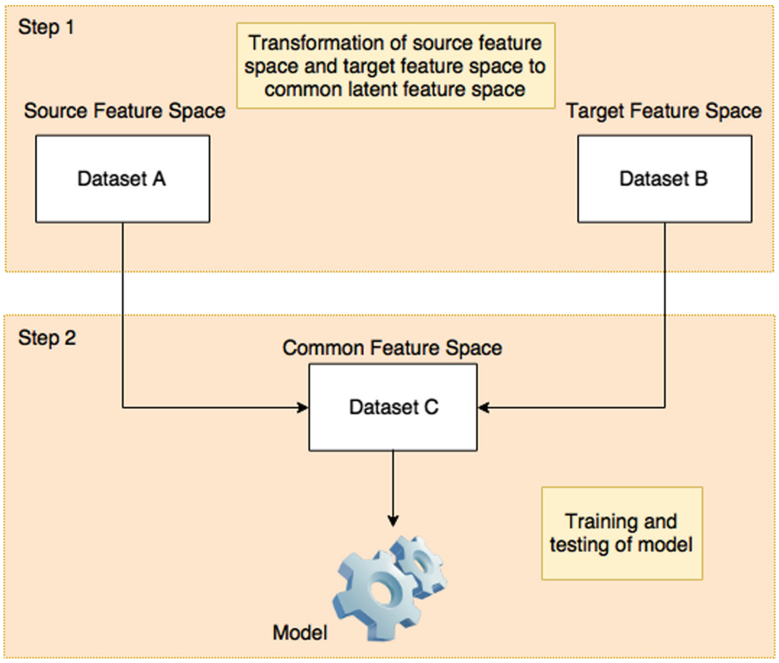

Step 1
Transformation of source feature space and target feature space to common latent feature space
Source Feature Space
Target Feature Space
Dataset A
Dataset B
Step 2
Common Feature Space
Dataset C
Training and testing of model
Model

Fig. 3. The symmetric approach is made up of two steps. Step 1 consists of mapping dataset A and dataset B to a common latent feature space. Step 2 consists of training and testing the model based on data from the common feature space.

reduction method called Maximum Mean Discrepancy Embedding [15]. Data from related but different domains is projected onto the common latent feature space. [5] transform the source and target features into a higher dimensional representation with source, target and common components. They also introduce an extension to this method which also works for unlabeled data [16]. [9] propose the Heterogeneous Spectral Mapping (HeMap) method. The main idea of HeMap is to find a common latent feature space for two heterogeneous tasks. A spectral mapping of the source and target feature spaces into a common latent feature space is designed. Spectral mapping is designed as an optimization task that maintains the original structure of the data while minimizing the difference between the two domains. This mapping adopts linear approaches such as rotation, scaling, permutation of row vectors, column vectors etc. to find a common latent feature space. [15] introduce Transfer Component Analysis for dimensionality reduction of features in a common latent space. A kernel-based feature mapping method has been introduced by [17]. This method maps the marginal distribution of data from different source and target domains into a common kernel space.

In this section we mentioned a few symmetric methods of solving heterogeneous transfer learning problems, but there exist more. We are not going to concern ourselves with them any further in this work, as our main focus is on asymmetric approaches. We recommend [4] for readers interested in a more extensive overview.

2.2 Asymmetric Feature-Based Approach

Asymmetric transformation is perceived as a new and unique approach alternative to symmetric transformation. It consists of finding transformations M_S, enabling us to map source domain data D_S to target domain data D_T with different feature spaces [18,19]. In practice we mostly encounter the following version of the problem: we have data from a source domain, which we map to data from a target domain using various techniques. After that we train a model (Fig. 4a - altered figure from [1]). We can also encounter a scenario where we have source and target data from the same domain, on which we trained a very good model. This model is successfully applied in production, e.g. deployed by the customer. However, due to various reasons, the target feature space changed and our model became unable to react to new data. In the case that we are not able to modify the model, we have to find an ideal mapping of target data to source while making sure that the error of the model doesn't change (Fig. 4b - altered figure from [1]). This approach poses a big challenge in the research area and as far as we know there are not many satisfying solutions, which would be fully automatic and domain independent. This article focuses on asymmetric heterogeneous approaches. All of Sect. 3 is dedicated to an overview of available methods.

3 Overview of Presented Solutions Using Asymmetric Approach

Asymmetric heterogeneous transfer learning finds application in several practical domains, where mainly methods of machine learning are employed. In this section we will introduce several basic research approaches to solving tasks related to a few areas which we indentified as the most popular for transfer learning. We start with the assumption that we have different source and target feature spaces, but the same task, as shown in Fig. 4. However, if we also had a different task, we would first make use of the methods of transfer between feature spaces followed by methods of homogeneous transfer learning for the transfer of knowledge between tasks (domains). For better orientation, all methods discussed in this section are described in Table 1.

We would like to make readers aware that the individual methods are classified into four main domains: Image Classification (Sect. 3.1), Cross-Language Classification (Sect. 3.2), Cross-Project Defect Prediction (Sect. 3.3) and Activity Recognition (Sect. 3.4). Methods described within these domains are independent of each other and were not evaluated against each other on data sets. Mainly due to the fact that some methods are not publicly accessible (as open-source repositories). We can find a comparison of the individual methods to some extent in [4].

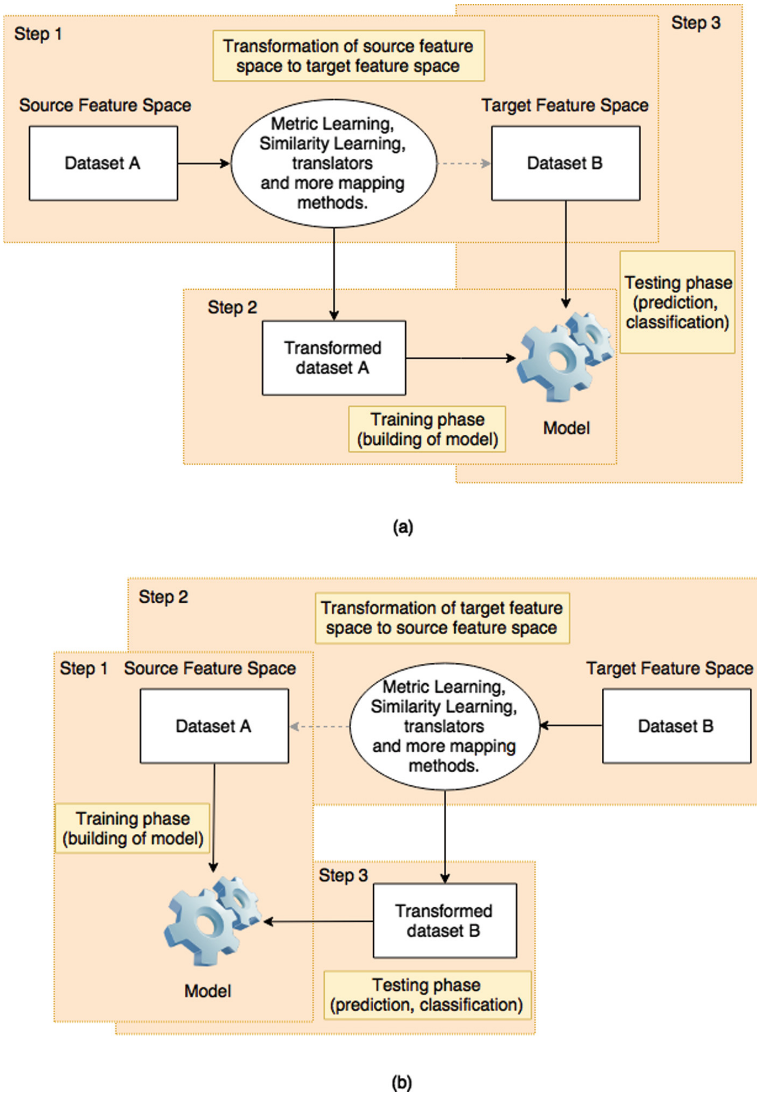

Step 1
Transformation of source feature space to target feature space
Source Feature Space
Target Feature Space
Dataset A
Metric Learning, Similarity Learning, translators and more mapping methods.
Dataset B
Step 3
Testing phase (prediction, classification)
Step 2
Transformed dataset A
Training phase (building of model)
Model
(a)
Step 2
Transformation of target feature space to source feature space
Step 1 Source Feature Space
Target Feature Space
Dataset A
Metric Learning, Similarity Learning, translators and more mapping methods.
Dataset B
Training phase (building of model)
Step 3
Transformed dataset B
Model
Testing phase (prediction, classification)
(b)

Fig. 4. There are two approaches to asymmetric transfer learning shown in (a) and (b). (a): Step 1 consists of mapping dataset A to dataset B using mapping methods. Step 2 consists of training the model based on data from the transformed dataset A, step 3 contains the phase where we test the model on dataset B from target feature space. (b): Step 1 consists of the training of the model based on data from dataset A from source feature space. In Step 2 we are looking for a mapping from dataset B to dataset A. Step 3 shows the testing of the model based on the transformed dataset B.

Table 1. Asymmetric heterogeneous transfer learning approaches discussed in Sect. 3.

Method	Source data	Target data	Transfer direction	Type of task
ARC-t	Labeled	Limited labels	S → T	Image classification
MOMAP	Labeled	Limited labels	S → T	Activity recognition
SHFR	Labeled	Limited labels	S ↔ T	Cross-language classification
HHTL	Labeled	Unlabeled	S ← T	Cross-language classification
van Kasteren's method	Labeled	Unlabeled	S → T	Activity recognition
MHTL	Labeled	Unlabeled	S → T	Activity recognition
FSR	Labeled	Limited labels	S ← T	Activity recognition, cross-language classification
TLRisk	Labeled	Unlabeled	S → T	Cross-language classification, image classification
HDP	Labeled	Unlabeled	S → T	Defect prediction
HMM	Labeled	Also unlabeled	S → T	Activity recognition

3.1 Image Classification

Image classification is very popular in the machine learning field. However it encounters many difficulties. An example of such difficulties can be labeling new images (requires a lot of human labor), ambiguous objects in images or objects having ambiguous explanations. Each image can also have different exposition, brightness, intensity, coloring or angle of view. Some of these problems can be solved by using transfer learning methods as will be shown below.

[18] solved the problem of domain adaptation for transferring object models from one dataset to another by introducing a novel approach in computer vision. The main idea is to learn an asymmetric non-linear transformation that maps data from one domain to another domain using supervised data from both domains. There is a requirement that the source data along with a part of the target data have to be labeled. The input consists of pairs of inter-domain examples that are known to be semantically similar or dissimilar. This approach can also be used in the case that some classes in the target domain have missing labels. [18] aims to generalize the model of [20] in his paper, which makes use of symmetric transformations. The new approach of [18] is called Asymmetric Regularized Cross-domain transformation, shortly ARC-t. ARC-t shows how a general formulation for the transfer learning problem can be applied in kernel space, resulting in non-linear transformations. The transformation matrix is learned in a non-linear Gaussian RBF kernel space. The resulting algorithm is based on squared Frobenius regularization [21] and similarity constraints. Similarity constraints are created for all pairs of data in the same class by using a similarity function. It helps us decide which pairs of data are similar and dissimilar. The

ARC-t method was tested using two experiments. The first experiment included 31 image classes in the source and target training data. In the second experiment only 16 image classes were included in the target training data and all 31 image classes were represented in the source training data. Within Kernel Canonical Correlation Analysis [22] the source and target domains were projected into common latent space using symmetric transformation. This preprocessing step is needed because the method needs to bring the source and target domains together to test against other baseline approaches. During testing the method showed certain advantages compared to existing baseline approaches such as k-nearest neighbors, SVM, domain and cross-domain metric learnings and the feature augmentation method proposed by [5]. The main idea of feature augmentation is to duplicate features in order to capture differences among domains. This method is briefly discussed in Sect. 2.1.

3.2 Cross-Language Classification

Another typical task in transfer learning is language transformation between multilingual documents or web pages, specifically the transformation from one language to another. This task can be solved using automatic translators mentioned in [23]. Transfer learning is aiming to solve the task without these tools, only using the transfer of knowledge between related feature spaces. Most methods learn a feature mapping between source and target domains based on data correspondences [18,19]. Correspondence means that there exist some relationships between data from different feature spaces.

[24] present a hybrid heterogeneous transfer learning (HHTL) framework. The method expects plenty of labeled source data and plenty of unlabeled target data. At first [24] learned an asymmetric transformation from the target to source domain. As a second step the common latent feature space using transformed data from the first step is discovered. Lastly, HHTL consists of deep learning which learns a feature mapping from the target domain to the source domain. HHTL simultaneously corrects the data bias on the mapped feature space. This framework was tested on multilingual text mining tasks and also in the application of text sentiment classification.

[19] propose to learn feature mapping based on the construction of feature correspondences between different feature spaces. This construction is called translator in [19]. In this work the language model is used. The language model is a probability distribution over sequences of words [25]. The language model links the class labels to the features in the source spaces and maps their translation to features in the target spaces. This novel framework is called translated learning (TLRisk), where training data and test data can come from totally different feature spaces. The main idea of translated learning is to translate all the data from source feature space into a target feature space. We assume there is no correspondence between instances in these different feature spaces. The language model proposed by [19] consists of feature translation and nearest neighbor learning. We can imagine this model as a chain of links which is modeled using a Markov chain and risk minimization. For the development of

a translator we need some co-occurrence data across source and target spaces. Performance of the TLRisk framework was shown on text-aided image classification and on cross-language classification (English to German classification). It is important to note that [19] interprets translated learning as a learning strategy different from transfer learning (other strategies: self-taught learning, multi-view learning, supervised learning etc.). There exist some similarities between transfer learning strategies and translated learning strategies [26], however translated learning is focused on more general and difficult problems than transfer learning. This is caused mainly by the representation of feature spaces - translated learning is looking for a translator between totally different spaces, for example the source feature space can be represented by a labeled text corpus and the target feature space by unlabeled images.

[27] proposed a domain adaptation method where data originates from heterogeneous feature spaces of different dimensions. The method includes a transformation matrix to map the weight vector of classifiers learned from the source domain to the target domain. This method works if the following requirements are met: sparse feature representation and class-invariant transformation. Sparse feature representation means that a feature in one domain can be represented by only several features in a different domain. The feature mapping across these domains is linear. Class-invariant transformation means that the feature mapping of some feature is invariant to different classes. To make the learning of a heterogeneous domain possible, the transformation matrix has to learn the similarity between source and target domain data. This data can be transformed from source feature space to target feature space and equivalently vice versa. [27] used the scheme - the Error Correcting Output Codes (ECOC) to generate binary classifiers for the multi-class classification problem [28]. With ECOC, their solution, called Sparse Heterogeneous Feature Representation (SHFR), can learn a transformation matrix. Part of the learning process of the transformation matrix is the adoption of a multi-task learning method based on [29]. Multi-task learning is based on learning more tasks simultaneously [3]. The SHFR method (also in combination with ECOC) was tested against DAMA [19], ARC-t [18] and HFA [12]. SVM classifier was used as a baseline method. The proposed method (SHFR) performed best for all tests.

3.3 Cross-Project Defect Prediction

Software defect prediction is another important application area in transfer learning and software engineering. There is a possibility to build a prediction model with defect data collected from a software project and predict defects for new projects (cross-project defect prediction - CPDP) [30–33]. However, projects must have the same metric set with identical meanings between projects.

[34] introduce heterogeneous defect prediction (HDP) which allows heterogeneous metric sets across projects. We consider one source and one target dataset, with heterogeneous metric sets. Each column and row represents a metric and an instance, the last column represents instance labels. Metric sets are not identical, each one contains different feature spaces. At first [34] apply feature selection

techniques to the source data. For feature selection they used various feature selection approaches widely used in defect prediction such as gain ratio, chi-square, relief-F, and attribute significance evaluation. After that, metrics based on the similarity such as correlation or distribution between the source and target metrics were matched up. With the final source dataset, heterogeneous defect prediction builds a model and predicts the labels of target instances. To match source and target metrics, [34] measure the similarity of each source and target metric pair by using several existing methods such as percentiles, Kolmogorov-Smirnov Test and Spearman's correlation. The key idea is computing matching scores for all pairs between the source and target metrics. Figure 5 (figure originally from [34]) shows the usage of a simple sample matching. There are two source metrics (X_1 and X_2) and two target metrics (Y_1 and Y_2). Thus, there are four possible matching pairs, (X_1, Y_1), (X_1, Y_2), (X_2, Y_1), and (X_2, Y_2). From all pairs between the source and target metrics, we remove poorly matched metrics whose matching score is not greater than a specific cutoff threshold. In this case the threshold is 0.3, so the X_1, Y_2 pair is not possible. After applying the cutoff threshold, they used the maximum weighted bipartite matching technique to select a group of matched metrics, whose sum of matching scores is the highest, without duplicates. The sum of matching scores in the first group is 1.3 (=0.8 + 0.5) and that of the second group is 0.4. Thus, we select the first matching group as the set of matched metrics. When mapping is finished, a model for target labels prediction is built using the mapped data. The HDP method uses logistic regression as the base learner. [34] used 5 groups with 28 datasets from the field of software defect prediction for their experiments.

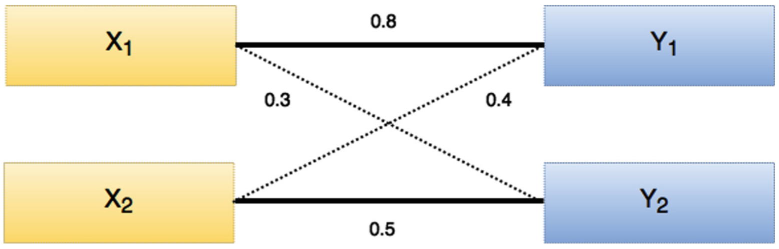


Fig. 5. The figure introduced in [34] shows the use of a simple sample matching between two source metrics (X_1 and X_2) and two target metrics (Y_1 and Y_2).

3.4 Activity Recognition

We are further going to state several examples from the activity recognition area where heterogeneous transfer learning finds numerous applications. In this area the activities of daily living are monitored through diverse sensors. This monitoring is crucial for the future of elderly care. There is motivation to use existing data from other houses (each house has different dispositions, uses different sensors etc.) in order to learn the parameters of a model for a new house.

The reason is that activity recognition models often rely on labeled examples of activities for learning, which are missing in a new house (this problem is known as the cold-start problem). Activity recognition and discovery models usually include information based on structural, temporal and also spatial features of the activities [7,35].

[35] also map sensors from source to target domain based on location or function. Their method is called Multi Home Transfer Learning (MHTL). MHTL composes of 3 phases - activity extraction, mapping and label assignment. The activity model consists of various features from sensors, such as structural, spatial and temporal. Their method is a good example of the utilization of meta-features. Meta-features are common for all data. It is a kind of mapping allowing us to have a single common feature space that can be used for all houses [36]. In [35] meta-features are first manually introduced into the feature space (for every source-target pair). Meta-features are features that describe properties of the actual features. This feature space is then automatically mapped from the source to target domain. Other works using meta-features are [36,37].

[36] use manual mapping strategies to group sensors based on their location or function. They proposed a method for applying transfer learning to time series data using a generative model. During training they are able to use both labeled and unlabeled data. Each sensor is described using one or more meta-features, for example the sensor on the washing machine might have one meta-feature specifying that the sensor is located in the bathroom, and another that the sensor is attached to the water pipes. The mapping of features occurs while learning the prior distribution, and using it to learn the target model parameters. [36] uses the Hidden Markov model to recognize activities and the EM algorithm to learn the model parameters of the target house. For the evaluation of the performance of the method they used three real world activity recognition datasets.

[38] designed a Multiple Outlook Mapping algorithm (MOMAP). MOMAP computes optimal affine mappings from different source feature spaces (in their terminology outlooks) to a target feature space by matching moments of empirical distributions. These spaces are not related through corresponding instances, but only through the common task. The optimal affine mapping is a function of geometric projection which maps the points lying on the same line onto one common point or to the same number of other points. Affine mapping preserves the division ratio [39]. In the MOMAP algorithm affine mapping is represented by translation and rotation. The mapping is done by matching moments of empirical distributions. The empirical distribution function is associated with an empirical measure of samples. Empirical measure means random measure realization of a sequence of random variables. The moments are quantiles from the empirical distribution function [40]. Before mapping in the MOMAP algorithm, the scaling of features of all spaces is required. The scaling aims to normalize the features of all spaces to the same range. Then mapping is performed by translating the means of each class to zero. Next, the rotation of the classes to fit each other is done using a rotation matrix. Finally, we have to translate the means of the mapped space to the final space. The framework performance is demonstrated on activity

recognition data taken from wearable sensors. The task consists of the classification of 5 different activities, where the source domain contains different but related sensor readings as compared to the target. The method was compared against a baseline method (SVM). MOMAP outperforms SVM in every test.

[8] propose a novel heterogeneous transfer learning technique called Feature-Space Remapping (FSR). FSR can handle the different feature spaces without the use of co-occurrence data (correlated data), as is shown for example in [19]. FSR maps the data to a different feature space using part of the labeled data from the target domain to infer relations to the source domain. FSR requires a one time manual specification of meta-features and then can be applied to map multiple source and target domains. [8] map the features from target feature space to a different source feature space. To achieve feature mapping, they learn a mapping from each dimension in the target feature space to a corresponding dimension in the source feature space. FSR computes the average similarity between the source and target meta-feature values for each pair between features from source and target feature spaces. The similarity is calculated between two meta-features as the absolute value of the difference between meta-feature values divided by the maximum possible difference between the meta-features. As a product we get many-to-one mapping. [8] evaluated the performance of FSR and its extensions in the activity recognition domain and in the document classification domain. By extensions they mean informed and uninformed FSR based on the availability of labeled data in the target domain. They evaluated 18 datasets from different smart apartments. A prerequisite were feature-rich datasets. Baseline comparison was provided by manual mapping and no mapping.

A lot of transfer learning methods solve situations where the difference between the source and target domains is caused mainly by differences in the marginal probability distributions of the domains [13,35,41]. By marginal probability distribution we mean probability distribution of the features contained in the subset of a collection of random features.

4 Feature Mappings

If we consider heterogeneous transfer learning, one of the main operations of transfer learning approaches is the mapping of features. These features can originate from different feature spaces. We can map the features into a common latent feature space (e.g. symmetric approach) or we can look for a mapping from one feature space to another (e.g. asymmetric approach). Finding an optimal mapping is a significant problem in the machine learning community. By mapping we mean a type of feature transformation (or combination of different types of transformations) which maps features from one feature space to another. The used transformation depends on the extent of the dissimilarity between mapped features. Standard types of transformation are translation or rotation. We have to also consider standard preprocessing methods between two different feature spaces such as data type conversion, normalisation or discretization.

We face many problems while mapping one feature space onto another: different number of features (e.g. different dimensionality of each feature space), different probability distribution, different data representation etc. In practice mapping of features is often done manually. However the number of possible mappings between source and target spaces grows exponentially as the number of features increases. In a lot of cases, manual mapping of features is impossible. We may also encounter the need to involve a domain expert. We can imagine a task from the medicine field where we would like to map two different datasets which come from different measurements of patient data from different machines. In these cases it is almost impossible to solve the mapping without the involvement of a domain expert. This requires both financial and human resources. The feature mapping method also depends on the existence of co-occurence data. If there is co-occurence data in our features space, there exist a lot of techniques of how to manage the mapping, see [42]. In the heterogeneous transfer learning field, no co-occurence data is common and no universal tool for their mapping is known. The selection of a suitable mapping depends on the type of data - numerical (each data instance is a numerical feature vector) or structured (each instance is a structured object such as a string, a tree etc.), and on the specific problem. We can also look for the mapping of each feature independently. We can distinguish data types as numerical and nominal. Nominal features are represented by labels or names and there is no ordering or distance measure among these values. Nominal features are represented by a finite number of categories. Numeric features are represented by real numbers.

If the number of feature combinations does not pose a combinatorial problem, it is possible to realize feature mapping in a manual way. [43] try to map activity sensors from two different flats (activity recognition task). They proposed a number of manual mapping strategies: intersect, duplicate, union. Intersect means that for each function group of sensors, they match similar sensors and sensors which have nothing in common are disregarded. Duplicate means that a matching of similar sensors is performed for each function group. Sensors that have no comparable sensor in the other house use any other sensor that exists in the same function group. Union means that the union of all the sensors in each function group is taken resulting in one sensor output per group per house. [43] They then experimentally determine which mapping strategy works best. However manual mapping can be domain dependent as is shown in [43]. And as we stated above, manual mapping is often combinatorically impossible.

State-of-art of feature mapping consists of preprocessing, dimensionality reduction and feature selection methods. There exist works which concern themselves with dimensionality reduction [13,15,44,45] and feature selection [46]. We can also encounter feature weighting methods. But a wide spectrum of the methods covers the preprocessing phase. In this work several mapping approaches used within transfer learning are presented. We can divide them into statistic and metric methods. Statistic methods are represented primarily by the Spearman's correlation coefficient, Kolmogorov-Smirnov test, Kullback-Leibler divergence etc. We will focus more on pairwise metrics which are based on measuring the distance or similarity between data. As stated above, each problem pertains to a specific domain which has its own semantic notion of data similarity. This data similarity can be difficult to express through standard metrics such as Minkowski metrics [47]. The solution seems to be learning the metric from data. This approach is called metric learning or similarity learning [21,47,48].

4.1 Metrics

There exist a lot of metrics, which can be used (tested) during the mapping of features. By metric we mean pairwise metric [47]. Pairwise metric is a way of measuring distance between objects. By distance we can also understand similarity. A fundamental method is the family of Minkowski distances including Euclidean, Manhattan and Chebyshev distances. We can also use cosine similarity to measure the cosine of the angle between two instances. It is widely used in data mining (e.g. bag-of-words or for sparse vectors). One of the most popular methods for comparing structured data is standard (Levenshtein) edit distance and its mutations (e.g. Specific Cost Matrix, tree or stochastic edit distance), where we search for the smallest number of transformations (insertion, deletion or substitution of symbols) needed to transform one string to another [47]. Many metrics exist as well as some good surveys [21,47], describing the individual metrics is out of the scope of this paper. Finding the correct metric basically solves the whole problem. There exist a lot of algorithms that rely on distances or similarities, for example k-Nearest Neighbors, Support Vector Machine, k-Means, information retrieval etc. However, finding a suitable metric can be a problem. Learning the metric from data seems to be a viable solution. This field is called similarity or metric learning.

4.2 Similarity and Metric Learning

Similarity learning is a subfield of supervised learning and its task is to discover a similarity function from an example that measures similarity between two objects. It is closely related to metric distance learning, which finds a distance function between data. These two areas are closely connected to transfer learning or domain adaptation, where we search for suitable methods of comparing the similarity of these features during the mapping of individual features between domains. We can then perform mapping based on these similarities which is represented by different transformations. The common process of metric learning can be seen in Fig. 6 (altered figure from [47]).

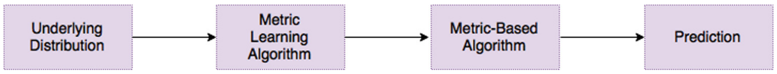


Fig. 6. Common process of metric learning.

We distinguish between linear (e.g. Mahalanobis distance learning and linear similarity learning) and nonlinear (e.g. kernelization of linear methods) metric learning [20,49]. The following survey on metric learning by [21,47,48] can serve for more details. Even though metric learning is a hot topic and is successfully used for problems in computer vision and other fields (e.g. [50–52]) as far as we know it is rarely used in the field of transfer learning and domain adaptation. Metric learning methods can also be divided into supervised and unsupervised according to the approach to learning. For supervised metric learning, a low-dimensional subspace is learned to preserve the geometrical information of the samples. For unsupervised metric learning, a discriminative distance metric is learned to maximize the separability of samples from different classes [53].

We can find one of few applications in the symmetric approach by [20] who present a method that adapts object models to new imaging conditions by supervised learning of transformations which minimizes the effect of domain-induced changes in the feature distribution. Supervised learning is used for learning of transformations and can be applied to categories which have no labeled examples in the new domain. This trend is similar to symmetric heterogeneous learning. However if a model is trained on one domain and then tested on another domain, it often results in poor performance [20]. One approach to this problem can be the generalization of the metric learning problem [18]. The idea is to learn a transformation A that maps the data from one domain to the other, thus leading to the inner product. This approach can be applied even when the dimensionalities of the two domains are different [21].

[52] presented a general discriminative method for learning similarity metrics. They propose a convolutional network for mapping data from source feature space to target feature space. Their method produces a non-linear mapping that can map any input vector of features to its corresponding version in lower dimension. It is also important that meta-features are learned from data and do not stem from prior knowledge about the task. The method can be used for recognition or verification applications where the number of categories is very large and not known during training, and where the number of training samples for a single category is very small [52].

[49] propose a novel metric algorithm to transfer knowledge from source to target feature space in metric settings called Cross-Domain Metric Learning (CDML). This method consists of three steps: (1) minimizing the distance between different distributions, (2) constructing two Gaussian distributions, one based on Mahalanobis distance to be learnt, second based on the information geometry [54] of target domain data, (3) constructing two more Gaussian distributions, one based on Mahalanobis distance again, the second one based on the labels and the geometry of source domain data. The results of these steps are combined into the unified loss function of CDML and by this combination the discriminating power gained from the labeled source domain data to the unlabeled target domain data is transferred.

Another usage of metric learning can be in unsupervised domain adaptation, where labeled source data and unlabeled target data are available for learning.

The aim is to unify source and target distributions. The solution can be the usage of a non-parametric way of measuring the distribution difference between the source and target samples called Maximum Mean Discrepancy (MMD) [47]. This is used by [55] in a domain adaptation metric learning (DAML) algorithm. Further we encounter a transfer metric learning (TML) approach by [56], where the metric and the task covariances between the source and target tasks are learnt under a unified convex formulation. Their work is based on multi-task metric learning with transfer learning settings.

5 Conclusion

Numerous transfer learning methods have been introduced in the past decade. In this survey we focused on heterogeneous transfer learning. Heterogeneous transfer learning can be divided into feature-based and instance-based methods according to what is being transferred. The feature-based methods gain more attention in the field, because the tasks are more complicated and pressing, with unclear solutions. The majority of feature-based approaches transform source and target feature spaces to a common latent feature space. This approach is called symmetric (see Sect. 2.1). The minority of works concern themselves with finding ideal mapping methods of source feature space to target feature space or vice-versa. This approach is called asymmetric (see Sect. 3). The asymmetric approach is significantly more demanding because we are looking for an optimal mapping between two different (but somehow related) feature spaces. We can consider a mappping which is possible in both directions (between source and target feature space and vice-versa) as optimal.

5.1 Challenges

The main contribution of this paper is to provide a summary of available up-to-date approaches and methods in the area of heterogeneous transfer learning. We also aim to emphasize some of the open challenges within this area. One basic formulation of the problem is: can the labeled data from other related sources help predict the target task, even if they have different feature spaces (e.g., image vs. text data) or different data distributions, or even different output spaces? Unfortunately as was described above, it depends on a lot of circumstances. A lot of the discussed methods are domain specific and the generalization of mapping transformations poses a challenge. In some cases generalization is impossible. Another challenge is constructing an automatic mapping of features between different feature spaces. Mapping is done manually in many cases and it is very time a human resources consuming. Automatic mapping will also have to face a challenge - there are two ways of automatic feature mapping: trying multiple mappings or mapping by analogy. This is often computationally very demanding. There are also some complications connected to the lack of overlap between feature spaces and different dimensionality. We also have to consider, whether there is any correspondence between features. One of the remaining questions is the negative transfer within asymmetric heterogeneous transfer learning and varying data [57]. Negative transfer represents a decision when transfer learning is still beneficial and when its use can have negative effects (e.g. is more demanding than manual mapping, model results have low accuracy etc.). The adaptation of metrics to varying data (e.g. lifelong learning, detection of concept drifts) may also pose a challenge.

Our future work will consist of finding suitable feature mappings between different source and target spaces. We would like to use these mapped features, more precisely the data, in machine learning models which were learnt on data not mapped and evaluate their relative performance. This paper forms a base for future work in the field of asymmetric heterogeneous transfer learning using methods of metric learning - this combination is not very common as far as we know and it is one of the main challenges which could bring an automatized and generalized solution for asymmetric heterogeneous transfer learning problems.

Acknowledgements. This research has been supported by SGS grant No. SGS17/210/OHK3/3T/18.

References

1. Friedjungová, M., Jiřina, M.: Asymmetric heterogeneous transfer learning: a survey. In: Proceedings of the 6th International Conference on Data Science, Technology and Applications - Volume 1: DATA, INSTICC, pp. 17–27. SciTePress (2017)
2. Zheng, Y.: Methodologies for cross-domain data fusion: an overview. IEEE Trans. Big Data **1**(1), 16–34 (2015)
3. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. Knowl. Data Eng. **22**(10), 1345–1359 (2009)
4. Weiss, K., Khoshgoftaar, T.M., Wang, D.: A survey of transfer learning. J. Big Data **3** (2016). https://doi.org/10.1186/s40537-016-0043-6. 9 pages
5. Daumé III, H.: Frustratingly easy domain adaptation (2009)
6. Bleiholder, J., Naumann, F.: Data fusion. ACM Comput. Surv. (CSUR) **41**(1), 1 (2009)
7. Cook, D.J., Feuz, K.D., Krishnan, N.C.: Transfer learning for activity recognition: a survey. Knowl. Inf. Syst. **36**(3), 537–556 (2013)
8. Feuz, K.D., Cook, D.J.: Transfer learning across feature-rich heterogeneous feature spaces via feature-space remapping (FSR). ACM Trans. Intell. Syst. Technol. **6**(1) (2015). https://www.ncbi.nlm.nih.gov/pmc/utils/ctxp/?ids=PMC4804893&report=citeproc&format=json. 3 pages
9. Shi, X., Liu, Q., Fan, W., Wu, P.S., Zhu, R.: Transfer learning on heterogeneous feature spaces via spectral transformation. In: 2010 IEEE 10th International Conference on Data Mining (ICDM). IEEE (2010)
10. Prettenhofer, P., Stein, B.: Cross-language text classification using structural correspondence learning. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (2010)
11. Wang, C., Mahadevan, S.: Heterogeneous domain adaptation using manifold alignment. In: IJCAI Proceedings-International Joint Conference on Artificial Intelligence (2011)

12. Duan, L., Xu, D., Tsang, I.W.: Learning with augmented features for heterogeneous domain adaptation (2012)
13. Blitzer, S.J., McDonald, R., Pereira, F.: Domain adaptation with structural correspondence learning. In: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics (2006)
14. Ando, R.K., Zhang, T.: A high-performance semi-supervised learning method for text chunking. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, ACL 2005, Stroudsburg, PA, USA, pp. 1–9. Association for Computational Linguistics (2005)
15. Pan, S.J., Kwok, J.T., Yang, Q.: Transfer learning via dimensionality reduction. In: Proceedings of the 23rd National Conference on Artificial Intelligence (2008)
16. Daumé III, H., Kumar, A., Saha, A.: Frustratingly easy semi-supervised domain adaptation. In: Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing. Association for Computational Linguistics (2010)
17. Zhong, E., Fan, W., Peng, J., Zhang, K., Ren, J., Turaga, D., Verscheure, O.: Cross domain distribution adaptation via kernel mapping. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM (2009)
18. Kulis, B., Saenko, K., Darrell, T.: What you saw is not what you get: domain adaptation using asymmetric Kernel transforms. In: 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2011)
19. Dai, W., Chen, Y., Xue, G.R., Yang, Q., Yu, Y.: Translated learning: transfer learning across different feature spaces. In: Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems (2008)
20. Saenko, K., Kulis, B., Fritz, M., Darrell, T.: Adapting visual category models to new domains. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part IV. LNCS, vol. 6314, pp. 213–226. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15561-1_16
21. Kulis, B., et al.: Metric learning: a survey. In: Foundations and Trends® in Machine Learning, Now Publishers, Inc. (2013)
22. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, New York (2004)
23. Wei, B., Pal, C.: Cross lingual adaptation: an experiment on sentiment classifications. In: Proceedings of the ACL 2010 Conference Short Papers (2010)
24. Zhou, J.T., Pan, S.J., Tsang, I.W., Yan, Y.: Hybrid heterogeneous transfer learning through deep learning. In: 28th AAAI Conference on Artificial Intelligence (2014)
25. Lafferty, J., Zhai, C.: Document language models, query models, and risk minimization for information retrieval. In: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM (2001)
26. Dai, W., Chen, Y., Xue, G., Yang, Q., Yu, Y.: Translated learning: transfer learning across different feature spaces. In: Advances in Neural Information Processing Systems 21, pp. 353–360. Curran Associates, Inc. (2009)
27. Zhou, J.T., Tsang, I.W., Sinno, P.J., Tan, M.: Heterogeneous domain adaptation for multiple classes. In: International Conference on Artificial Intelligence and Statistics (2014)
28. Dietterich, T.G., Bakiri, G.: Solving multiclass learning problems via error-correcting output codes. J. Artif. Intell. Res. **2**, 263–286 (1995)
29. Ando, R.K., Zhang, T.: A framework for learning predictive structures from multiple tasks and unlabeled data. J. Mach. Learn. Res. **6**, 1817–1853 (2005)

30. Nam, J., Pan, S.J., Kim, S.: Transfer defect learning. In: Proceedings of the 2013 International Conference on Software Engineering. IEEE (2013)
31. He, Z., Shu, F., Yang, Y., Li, M., Wang, Q.: An investigation on the feasibility of cross-project defect prediction. Autom. Softw. Eng. **19**(2), 167–199 (2012)
32. Ma, Y., Luo, G., Zeng, X., Chen, A.: Transfer learning for cross-company software defect prediction. Inf. Softw. Technol. **54**(3), 248–256 (2012)
33. Rahman, F., Posnett, D., Devanbu, P.: Recalling the "imprecision" of cross-project defect prediction. In: Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering. ACM (2012)
34. Nam, J., Kim, S.: Heterogeneous defect prediction. In: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering. ACM (2015)
35. Rashidi, P., Cook, D.J.: Multi home transfer learning for resident activity discovery and recognition (2010)
36. van Kasteren, T.L.M., Englebienne, G., Kröse, B.J.A.: Transferring knowledge of activity recognition across sensor networks. In: Floréen, P., Krüger, A., Spasojevic, M. (eds.) Pervasive 2010. LNCS, vol. 6030, pp. 283–300. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12654-3_17
37. Blanke, U., Schiele, B.: Remember and transfer what you have learned - recognizing composite activities based on activity spotting. In: 2010 International Symposium on Wearable Computers. IEEE (2010)
38. Harel, M., Mannor, S.: Learning from multiple outlooks. In: Proceedings of the 28th International Conference on Machine Learning (2011)
39. Nomizu, K., Sasaki, T.: Affine Differential Geometry, 1st edn. Cambridge University Press, Cambridge (1995)
40. van der Vaart, A.W.: Asymptotic Statistics, 1st edn. Cambridge University Press, Cambridge (2000)
41. Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., Wortman, J.: Learning bounds for domain adaptation. In: Advances in Neural Information Processing Systems 21. MIT Press (2007)
42. Cook, D.J., Krishnan, N.C.: Activity Learning, Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. Wiley, New York (2015)
43. van Kasteren, T.L.M., Englebienne, G., Krose, B.J.A.: Recognizing activities in multiple contexts using transfer learning. In: AAAI Fall Symposium: AI in Eldercare: New Solutions to Old Problems (2008)
44. Si, S., Tao, D., Geng, B.: Bregman divergence-based regularization for transfer subspace learning. IEEE Trans. Knowl. Data Eng. **22**(7), 929–942 (2010)
45. Dai, W., Jin, O., Xue, G.R., Yang, Q., Yu, Y.: EigenTransfer: a unified framework for transfer learning. In: Proceedings of the 26th Annual International Conference on Machine Learning. ACM (2009)
46. Satpal, S., Sarawagi, S.: Domain adaptation of conditional probability models via feature subsetting. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 224–235. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74976-9_23
47. Bellet, A., Habrard, A., Sebban, M.: A survey on metric learning for feature vectors and structured data (2013)
48. Yang, L.: Distance metric learning: a comprehensive survey (2006)
49. Wang, H., Wang, W., Zhang, C., Xu, F.: Cross-domain metric learning based on information theory. In: 28th AAAI Conference on Artificial Intelligence (2014)
50. Chechik, G., Sharma, V., Shalit, U., Bengio, S.: Large scale online learning of image similarity through ranking. J. Mach. Learn. Res. **11**, 1109–1135 (2010)

51. Kulis, B., Jain, P., Grauman, K.: Fast similarity search for learned metrics. IEEE Trans. Pattern Anal. Mach. Intell. **31**(12), 2143–2157 (2009)
52. Chopra, S., Hadsell, R., LeCun, Y.: Learning a similarity metric discriminatively, with application to face verification. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE (2005)
53. Hu, J., Lu, J., Tan, Y.P., Zhou, J.: Deep transfer metric learning. IEEE Trans. Image Process. **25**(12), 5576–5588 (2016)
54. Wang, S., Jin, R.: An information geometry approach for distance metric learning. In: Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (2009)
55. Geng, B., Tao, D., Xu, C.: DAML: domain adaptation metric learning. IEEE Trans. Image Process. **20**(10), 2980–2989 (2011)
56. Zhang, Y., Yeung, D.Y.: Transfer metric learning by learning task relationships. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM (2010)
57. Day, O., Khoshgoftaar, T.M.: A survey on heterogeneous transfer learning. J. Big Data **4**(1), 29 (2017)

A Mathematical Model for Customer Lifetime Value Based Offer Management

Ahmet Şahin[1](✉), Zehra Can[1,2](✉), and Erinc Albey[1](✉)

[1] Department of Industrial Engineering, Ozyegin University, Istanbul, Turkey
{ahmet.sahin, zehra.can}@ozu.edu.tr,
erinc.albey@ozyegin.edu.tr

[2] Turkcell Technology Research and Development Inc., Istanbul, Turkey

Abstract. Customers with prepaid lines possess higher attrition risk compared to postpaid customers, since prepaid customers do not sign long-term obligatory contracts and may churn anytime. For this reason, mobile operators have to offer engaging benefits to keep prepaid subscribers with the company. Since all such offers incur additional cost, mobile operators face an optimization problem while selecting the most suitable offers for customers at risk. In this study, an offer management framework targeting prepaid customers of a telecommunication company is developed. Proposed framework chooses the most suitable offer for each customer through a mathematical model, which utilizes customer lifetime value and churn risk. Lifetime values are estimated using logistic regression and Pareto/NBD models, and several variants of these models are used to predict churn risks using a large number of customer specific features.

Keywords: Customer lifetime value · Offer management · RFM
Pareto/NBD · Logistic regression · Mobile subscribers · Prepaid subscribers
Telecommunication

1 Introduction

Mobile operators have to adapt to digital evolution of the customer services. Smartphones has changed the market conditions. Customers can experience hundreds of digital services and applications, easily, with their smartphones. Therefore, operators cannot be solely seen as communication providers anymore. They have to provide profitable services to survive in the business and to improve the customer experience to prevent customer churn.

In fact, mobile operators would prefer postpaid subscribers over prepaid subscribers, since it is easier to manage the postpaid subscriber's behavior and the revenue stream generated by a postpaid subscriber is more predictable (and most of the time higher) than that of a prepaid subscriber. But in today's digital marketing ecosystem, telecommunication operators have to act not only a call service provider (CSP), but also act like a full service provider (OTT – Over the top provider) who offers many interactive social service products for their customers. Thus, keeping a customer is more valuable than before which means more users for the OTT services. These services are also an important means for revenue generation from the prepaid

J. Filipe et al. (Eds.): DATA 2017, CCIS 814, pp. 27–45, 2018.
https://doi.org/10.1007/978-3-319-94809-6_2

customers. Thus analyzing the churn behavior of prepaid customers as wells as their lifetime values has higher importance more than ever in the telecommunications sector.

The service and products of the operator should comply with the customer's needs. If this is not the case, it is highly probable that customers may leave and never come back or may never subscribe for a service of the operator. Customer experience management starts with potential customers' perception about the company brand. Starting with first impression, customer experience management aims to understand the desires of customers and to provide them with easy, simple and seamless experience with the offered products and services. The higher the quality in customer experience, the more customer will be attracted to and stay with the company.

Retaining customers has always been a costly challenge. Companies have developed different strategies to fight against customer attrition. Most of these efforts include proposing offers in varying forms, such as discounts, extra benefits etc. Almost in all of these efforts, increasing the amount of customer specific information pays off. In the digitalization era, the amount of customer specific data is more than ever, which yields numerous opportunities for developing granular and customized prediction models. When customer retention is considered, the challenge is turning this massive amount of data (and models built on the data) into business insights and eventually generate actions that decrease the retention cost, while increasing profit.

Churn analysis is one of the most fundamental customer behavior analysis in identifying silent, unhappy customers. The main output of churn analysis is a risk score, which quantifies the likelihood of customer churn. The output of churn analysis is best utilized when it is supported with extra customer specific features such as customer lifetime value (CLV). The relationship between churn and CLV is depicted in Fig. 1 which is based on the idea in the [1].

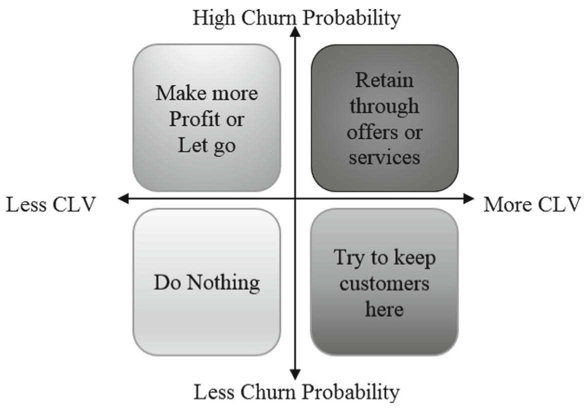

High Churn Probability
Make more Profit or Let go
Retain through offers or services
Less CLV
More CLV
Do Nothing
Try to keep customers here
Less Churn Probability

Fig. 1. Churn and CLV relationship.

As can be seen in Fig. 1, the ideal position of customers for a company is the bottom right, where customers produce the highest profit with lowest churn risk. It is recommended that companies should give special attention (provide offers or new exciting services) to the customers falling in the upper right region. Since, if retained in the company, these are the "high profit promising" customers. Bottom left and upper left regions are the regions, where less profitable customers reside. If possible, companies should try to increase profit made out of these customers. On the other hand, if the customer is not promising higher profit and possessing high churn risk, companies may simply let these customers go and try to acquire new customers, who fall into high CLV, low churn risk region.

Companies usually identify risky customers according to the lift of their favorite "churn-prediction" model. In a typical setting, they try to give the best offer to the customers possessing highest risk. In this scenario, managing the campaign is simple but its efficiency (in terms of cost) is questionable. Because, in such an execution, decision maker does not know whether offer receiving customers are worth the effort or not. On the other hand, integrating CLVs into the offer management process helps to increase return of the campaign by targeting, and possibly retaining, customers with higher CLV.

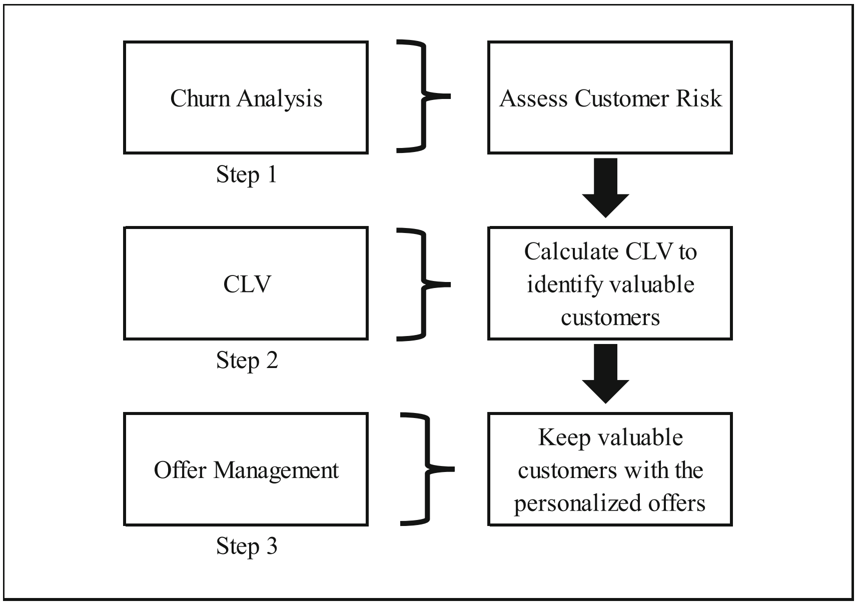

Churn Analysis
Step 1
Assess Customer Risk
CLV
Step 2
Calculate CLV to identify valuable customers
Offer Management
Step 3
Keep valuable customers with the personalized offers

Fig. 2. 3-level offer management process.

Figure 2 shows our proposed 3-level offer management framework. The first step calculates churn probability of customers, whereas the second step estimates CLVs. In the third step, an offer management optimization model is solved using the estimated churn and CLVs from the first two steps.

In our previous study [2] we focused on churn prediction models for prepaid subscribers of a mobile operator. This study can be seen as an extension, where we improve the performance of our churn prediction model, and include a CLV prediction approach. The major contribution of this paper is the proposed offer management model, which considers the predicted churn and CLV values for each subscriber and decide the best offer for each customer such that the expected overall CLV of the entire subscriber set is maximized where offers are addressed to subscribers considering budget and several offer allocation limitations.

The rest of the paper is organized as follows. Section 2 provides some background information on CLV and offer management (specifically in telecommunication sector). Section 3 overviews the literature. Section 4 introduces the data set used in this study. Section 5 presents performance comparison several logistic regression models for churn prediction, whereas Sect. 6 presents performance of Pareto/NBD model on churn and CLV prediction. Section 7 presents the proposed offer management model and discuss solutions of the optimization model for varying scenarios.

2 Background

2.1 Customer Lifetime Value (CLV)

CLV can be simply described as net the present value of the future cash flows associated with a customer [3]. If a mobile operator wants a profitable customer management they have to calculate the CLV of a subscriber [1] and a profitable customer means loyalty. Loyal customers spread positive word of mouth about the brand if they feel satisfied with the product and/or services. The studies on customer life time value in literature can be classified in four main categories [4] as shown in Table 1:

Table 1. CLV models [4].

Focus	Category	Model
Structural model	Customer unit	Individual model
		Segment (Customer base) model
	Prediction data	Retrospective model
		Prospective model
	Transaction	Contractual model
		Non-contractual model
	Purchase cycle	Discrete model
		Continuous model
Strategic model	Strategic use of CLV in management	
Normative model	Relationship between duration and cost	
Analytic model	Resource allocation (Budget allocation)/Pricing	

In this study we focus on both “Non-Contractual model” and “Discrete model” settings. In [5], the assumption monetary value is independent of the underlying transaction process, helps to decompose the CLV estimation into two sub-problems, namely transaction flow determination and monetary value estimation for transactions. A sub-model for the transaction flow (recency, frequency); discounted expected transaction is calculated based on the Pareto/NBD parameters, the confluent hypergeometric function of the second kind, and the Pareto/NBD likelihood function. The second sub-model for revenue per transaction (expected average transaction value); is calculated as a weighted average of the population mean and the observed average transaction value, which is the average value across transactions.

2.2 Offer Management

In the first years of mobile communication, a very limited number of customer services (only voice and short message services) were available. As mobile phone technology evolved, GSM service providers improve the quality and variety of services they provide and adapted to the rapidly changing technology faster than other industries. If subscribers are not satisfied with the service they receive or they think that the value proposed by the operator is minimal, they can easily switch their mobile service providers. Therefore, during the years, mobile operators generated various products and services for their subscribers to retain them with the company. Mobile operators created their own product lines in a very broad perspective and each product can easily be seen as an offer to the customer. In addition to this, offers also include marketing activities to attract and acquire customers over multiple channels. All these activities and offers are analyzed under the name campaign management (or alternatively offer management), which mainly aims to contact with the right customers at the right time over the right channel with right offer. Figure 3 overviews the general campaign management cycle.

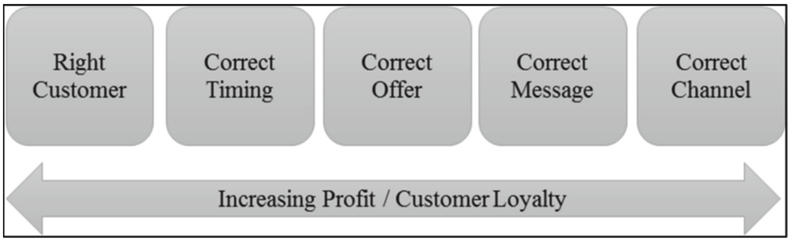


Fig. 3. Campaign/Offer management cycle.

A sample offer management framework for a mobile operator is presented in Fig. 4. The data is collected in “Enterprise Data Warehouse” (EDW) and also the mining models are fed from the EDW. Both the score from the mining models and the EDW data are used to calculate the best offer for each subscriber. The offer management system controls the calculated offers to propose to the subscribers. Integrated Voice Response (IVR), Short Message Service (SMS) and Web channels are different channels used in the offer management systems.

The Average Revenue per User in Turkish Lira (TL) (ARPU-TL) and subscriber trend is shown in Figs. 5 and 6, respectively [2]. This market data is published

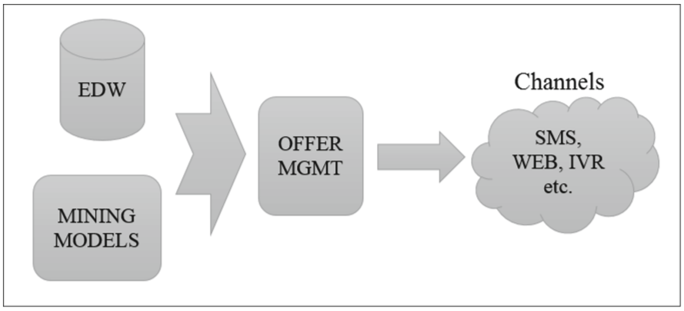


Fig. 4. Offer management framework.

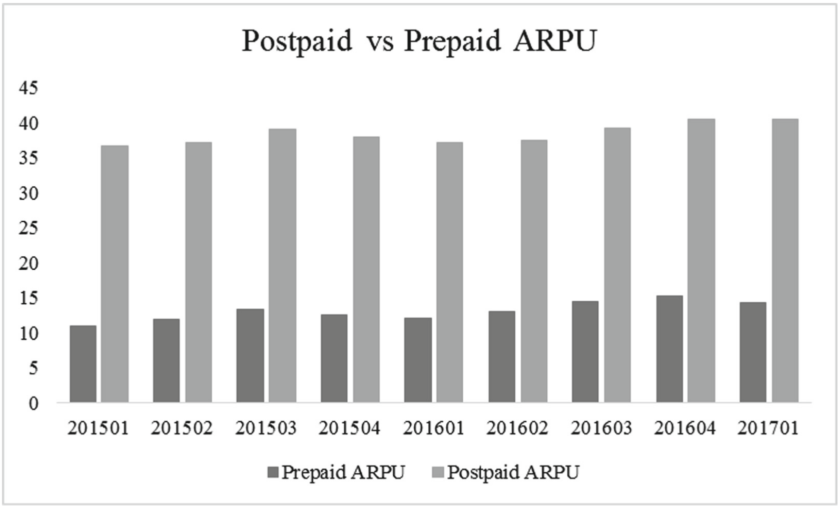


Fig. 5. ARPU trend for postpaid & prepaid subscribers.

quarterly to report the market trends in mobile sector in Turkey by “Bilgi Teknolojileri Kurumu” (BTK) which is the Governmental Organization of Information Technologies. It can be seen that ARPU of the postpaid subscribers is higher than the prepaid subscribers and the time shows that the number of the postpaid and prepaid subscribers converge to each other. It is also known that keeping a subscriber is cheaper than acquiring a new customer. If a mobile operator calculates the CLV of the prepaid subscriber and propose them the right offer than it is obvious that the operator can increase their revenue.

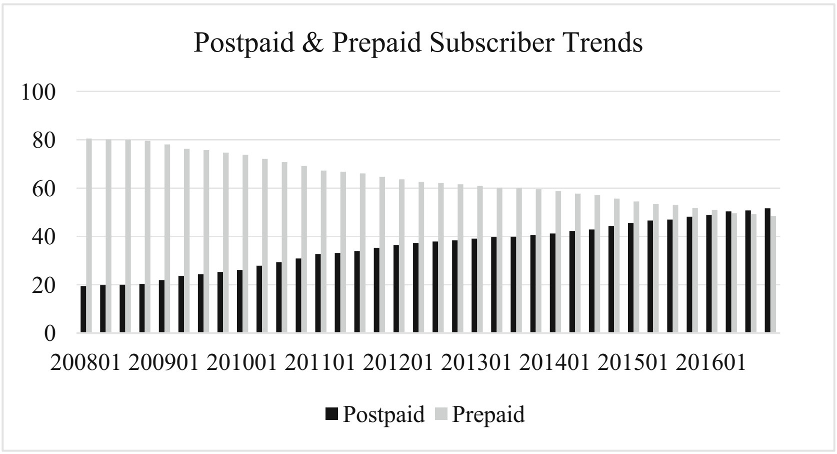


Fig. 6. Postpaid & prepaid subscribers trends.

3 Related Works

Recency-frequency-monetary (RFM) data is considered as one of the major set of features in segmenting customers. Our study uses refill history of prepaid subscribers which can be seen as a purchase order thus can be easily converted to *recency-frequency-monetary* (RFM) data. Details RFM applications in telecommunication sector can be found in [6], which emphasizes that although using RFM for customer analytics is a simple yet a strong approach, there are also some disadvantage, such as:

- RFM only focuses on best customers. If customer do not buy often or spend little or do not generate any transaction lately, it provides little information,
- RFM only focuses on limited variables. It is better to take into account other customer relation variables along with RFM data.

Although it has some disadvantages, RFM data still being frequently used by analysts due to its simplicity. Besides analytics activities for telecommunication sector, RFM has been applied to many other areas and activities [7]. One specific example is using RFM data for calculating CLVs. The calculation of customer based CLV is not a very frequently studied topic for mobile operator companies. However, there are a lot of churn analysis studies in the telecommunication analytics literature.

A churn analysis that is not backed up by a CLV analysis may not always provide the correct insights about customers. Although there are only a few studies which cover CLV calculation for telecommunications industry, these studies do not consider prepaid mobile subscribers. The study [4] deals with wireless telecommunication subscribers and calculate the CLV based on a new approach of Markov chain model. In [8], authors conduct a churn prediction study for land line subscribers, whereas [9] is focused on contracted telecommunication services (land-line or mobile phone or internet line). In [9], several hypotheses are defined in terms of validation and calculation of the CLV and customer equity values.

CLV calculation cannot be ignored in todays' challenging marketing. In the big data era, making descriptive analysis on historical data is not enough for improving customer loyalty. The competition conditions in any market compelled the companies to analyze the future behavior of their customer to improve the customer experience with their product. Thus, the CLV calculation is a "must" in todays' market. Especially, like mobile market where the prepaid subscribers almost constitute the 50% of the customer base (see Fig. 6), the mobile companies have to focus analyzing their prepaid subscribers, who have more complicated behaviors (harder to predict) than that of postpaid subscribers. As mentioned previously we utilize CLV information to allocate the best possible offer to subscribers in order to retain them with the company. Especially in the context of offer management, inclusion of CLVs play a critical role [10], hence the developed mathematical model is designed to reflect the effect of proposed offers on the CLVs.

The RFM data was extended by [5] for the CLV calculation. As stated in the paper, the study focused on using only the RFM data for the CLV calculation in customer base. The customer base data is the transactions for the non-contractual setting. This CLV calculation based on RFM data can be easily adopted to prepaid subscribers in mobile sector. However, there is no application of this method for mobile prepaid subscribers, we can give some other applications of CLV calculation based on RFM data. This method is used widely in many industries, there are many case studies like retail in these papers [11, 12].

In [4], the Pareto/NBD model is used for conducting customer base analysis. Pareto/NBD model is used to define the non-contractual purchase transactions for customers, for whom the actual churn time is never know with 100% certainty. There are many other models that can be used instead of Pareto/NBD model. In [12] a comparison for different models are provided.

Offer management is also another aspect of controlling churn of subscribers. A mobile operator has to manage their cost for customer loyalty. In [1], it is emphasized that companies cannot just only focus on profitable customer but also finding the loyal customers is more important than profitable customers. It is also important that companies have to solve the optimization problems for the promotions and calculate the maximum profit from their offer campaigns [13].

4 Data Set

The lifecycle period for the prepaid subscriber is 270-days. If they do not do any refill action, after 270-days the subscription will be ended by the mobile operator. But this period will be reset by refill transaction and another 270-days new period starts.

The total subscriber used in this study is 6480 which were activated two-year period starting from 1.2.2015 to 31.1.2017. Refill transactions are also for the same period for the activated subscribers. The subscribers are chosen among prepaid subscribers who did not do any charging method change and remains prepaid subscriber for their lifetime.

In this study we consider a three mutually exclusive segments, which are named as *youth*, *mass*, and *other*. Youth segment contains subscribers under age 26, and *other* segment is composed of subscribers for whom company has very limited information, and *mass* segment contains all other subscribers who are neither in *youth* nor in *other* segment. The distribution of subscribers over segments can be seen in Table 2.

For the RFM data, the refill transactions belong to again the same two-year period with the subscribers. For recency parameter we have taken the refill date, for the monetary value we have taken the refill amount in TL. The number of rows is 53.249. The # of distributions can be seen in Table 2 below.

Table 2. The distribution of subscribers over segments.

Segment information	Other	Youth	Mass	Total
No. of subscriber	1.063	1.352	4.065	6.480
No. of refill transactions	6.290	12.620	34.338	53.248

5 Churn Prediction Using Logistic Regression

In this section we present a comparative analysis for the performances of several Logistic Regression models for churn prediction. The base data set is assumed as the RFM data. In addition to this data set, we consider additional 395 features, which are collected by the company. The additional data set includes different types of features such as:

- call or usage behavior, includes incoming or outgoing calls and the types of communication like voice, data or short message service (SMS),
- network metrics, include call drop rates,
- refill metrics, include the behavior of the refill transactions of the subscriber like refill amount and also usage type based refill utilization like if the subscriber uses mostly SMS, data or voice,
- ARPU, includes the average revenue of per user in TL amount,
- number of lines, shows whether the subscriber has other lines, or has one or more lines,
- age,
- value-added-services (VAS) usage/subscription, indicates the subscriber's usage behavior of value added service or if the subscriber has any subscription for the value added services,
- pre-calculated social network analytics (SNA) metrics,
- network type (2G, 3G, 4G), includes the network type of the subscriber,
- equipment type, give if the subscriber has a smartphone or other basic phones.

Feature set selection is executed as follows:

- 395 features are analyzed with statistical methods,
- the features whose minimum and maximum values are equal are eliminated. 383 features are left,
- correlation analysis is run and 308 features are left,

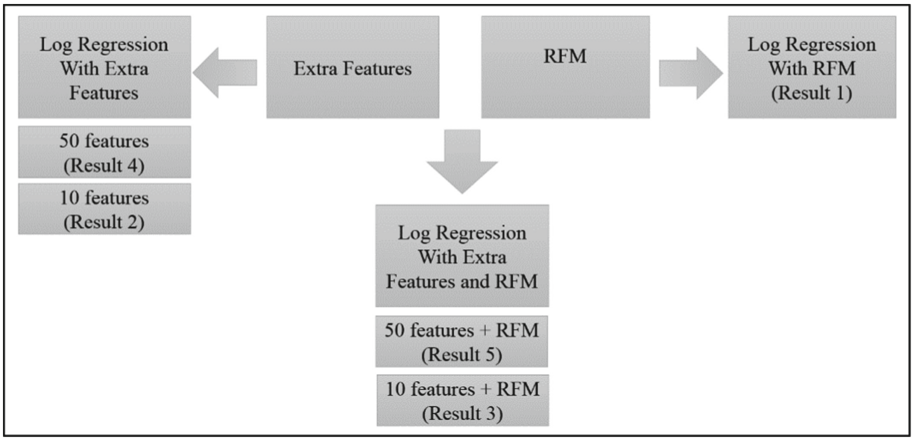

Log Regression With Extra Features
Extra Features
RFM
Log Regression With RFM (Result 1)
50 features (Result 4)
10 features (Result 2)
Log Regression With Extra Features and RFM
50 features + RFM (Result 5)
10 features + RFM (Result 3)

Fig. 7. Flow of benchmarking.

- with domain knowledge and expertise, the total number of features is reduced to 50,
- Stepwise regression with backward elimination is run over the 50 variables and 10 features are selected.

Logistic Regression is applied in the following order (also shown in Fig. 7):

- Model 1: Logistic Regression with RFM
- Model 2: Logistic Regression with 10 features
- Model 3: Logistic Regression with 10 features and RFM
- Model 4: Logistic Regression with 50 features
- Model 5: Logistic Regression with 50 features and RFM

The results for all combinations are shared in Table 3. Table 3 contains the following columns:

- cutoff value, which is used as a threshold to decide whether customer considered as a churner or non-churner combination,
- result, indicates the combination name
- set name, shows the details of the combination (selected data set),
- accuracy, true positive (TP), false positive (FP) rates and precision are the classical performance indicators for binomial classifiers, such as logistic regression.

The results presented in Table 3 reveals that base logistic regression model using only RFM data performs very well (in terms of accuracy and prediction). However, inclusion of extra customer specific variables results in a performance increase. Overall, the cutoff rate 0.9 yields the best result and the top performing models at this cutoff level are Model 3 and Model 5. However, if one model is to be selected, it should be Model 3 due to the rule of parsimony.

Table 3. Benchmark of data with logistic regression.

Cutoff	Model	Set name	Accuracy	TP rate	FP rate	Precision
0.5	Model 1	RFM	95%	98%	24%	96%
	Model 2	Var 10	91%	100%	64%	90%
	Model 3	Var 10 + RFM	95%	100%	31%	95%
	Model 4	Var 50	92%	100%	56%	91%
	Model 5	Var 50 + RFM	96%	100%	28%	95%
0.6	Model 1	RFM	94%	97%	22%	96%
	Model 2	Var 10	92%	100%	50%	92%
	Model 3	Var 10 + RFM	96%	100%	24%	96%
	Model 4	Var 50	93%	99%	41%	93%
	Model 5	Var 50 + RFM	96%	100%	23%	96%
0.7	Model 1	RFM	93%	95%	20%	96%
	Model 2	Var 10	94%	99%	34%	94%
	Model 3	Var 10 + RFM	97%	100%	20%	97%
	Model 4	Var 50	95%	99%	30%	95%
	Model 5	Var 50 + RFM	97%	100%	18%	97%
0.8	Model 1	RFM	92%	93%	17%	97%
	Model 2	Var 10	94%	98%	29%	95%
	Model 3	Var 10 + RFM	97%	99%	15%	97%
	Model 4	Var 50	95%	98%	24%	96%
	Model 5	Var 50 + RFM	97%	99%	14%	98%
0.9	Model 1	RFM	86%	86%	13%	97%
	Model 2	Var 10	95%	98%	24%	96%
	Model 3	Var 10 + RFM	98%	99%	12%	98%
	Model 4	Var 50	95%	98%	20%	97%
	Model 5	Var 50 + RFM	98%	99%	11%	98%

6 Churn and CLV Prediction Using Pareto/NBD Model

In preceding study [2], we focused on subscriber refill transactions and did not include any tenure or segment information of the subscriber in the transaction data. We found out that without any tenure or segment information the Pareto/NBD model fails in the test period while showing a good performance in the training period.

Our segment information includes youth, mass, and other (not segmented); definitions of which were provided in Sect. 4. First we generated the results based on the segment information then included 8 tenure groups which were based on the 3-months sub period information. The tenure is calculated according to the first refill action date. Segmentation based on tenure idea is illustrated in Fig. 8.

The Pareto/NBD model is executed based on the data sets, details of which is described above. The results are shared in Table 4. The Pareto/NBD model results is labeled as Model 6, Model 7, Model 8 and details of labeling is as follows:

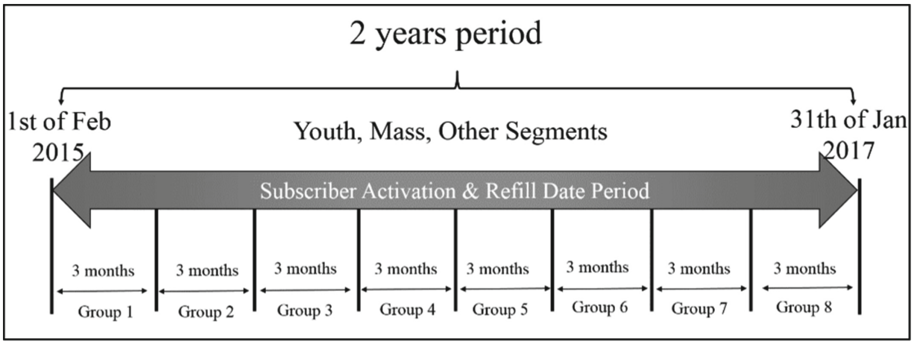


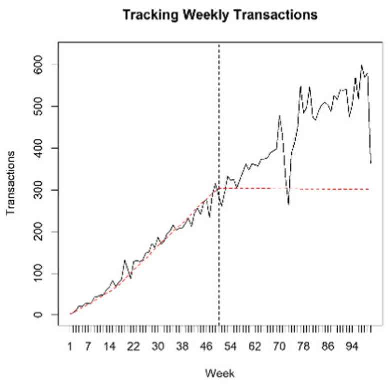
Fig. 8. Tenure groups.

Table 4. Pareto/NBD model results.

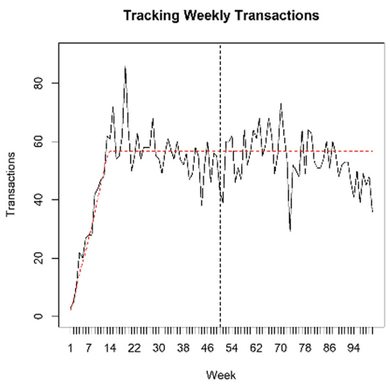
Cutoff	Model	Set name	Accuracy	TP Rate	FP Rate	Precision
0.5	Model 6	w/o Segment + Tenure	88%	98%	69%	89%
	Model 7	Segment	95%	99%	29%	95%
	Model 8	Segment + Tenure	94%	98%	29%	95%
0.6	Model 6	w/o Segment + Tenure	88%	97%	64%	90%
	Model 7	Segment	96%	99%	18%	97%
	Model 8	Segment + Tenure	96%	98%	18%	97%
0.7	Model 6	w/o Segment + Tenure	89%	97%	58%	91%
	Model 7	Segment	96%	98%	13%	98%
	Model 8	Segment + Tenure	96%	97%	13%	98%
0.8	Model 6	w/o Segment + Tenure	90%	96%	46%	92%
	Model 7	Segment	97%	98%	10%	98%
	Model 8	Segment + Tenure	96%	97%	10%	98%
0.9	Model 6	w/o Segment + Tenure	91%	93%	18%	97%
	Model 7	Segment	97%	97%	6%	99%
	Model 8	Segment + Tenure	96%	96%	6%	99%

- Model 6: Pareto/NBD model is run with RFM without segment and tenure information,
- Model 7: Pareto/NBD model is run with RFM with segment information,
- Model 8: Pareto/NBD model is run with RFM with segment and tenure information.

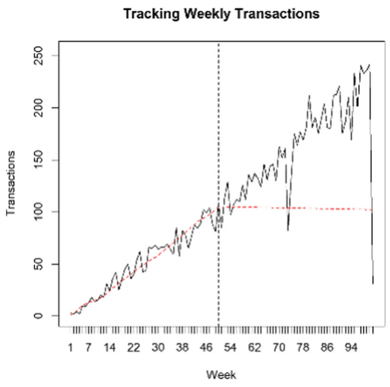
The Pareto/NBD model's performance in predicting expected number of transactions is given in Fig. 9a–f for the all set, mass, youth and not segmented sets. For each segment, the prediction power is tested under two scenarios. In the first set of scenarios the customers are segmented further with respect to their tenure; where as in the second case customers are further sub-segmented using their tenure information. The segment & tenure based data sets have the best performance for the Pareto/NBD model.



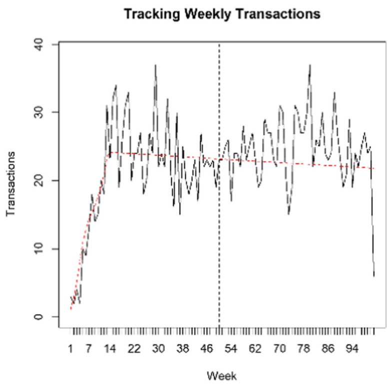
a. Mass segment



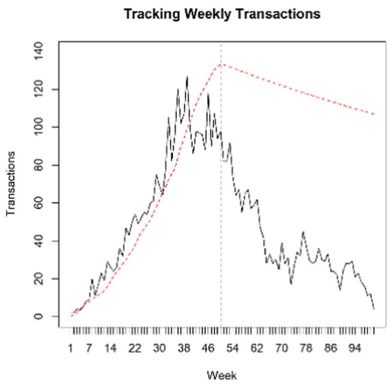
b. Mass Segment & Tenure



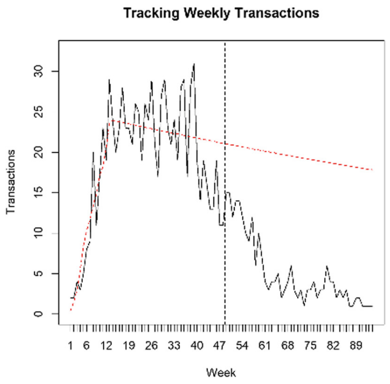
c. Youth Segment



d. Youth Segment & Tenure



e. Others Segment



f. Others Segment & Tenure

Fig. 9. Pareto/NBD model prediction performance for varying segment and tenure combinations.

In this study by adding the tenure and segment information we have got a good performance for the model in test period for that the expected number of transactions and probability of alive (one minus probability of churn) of each subscriber are calculated and used in the proposed offer management model.

7 Proposed Mathematical Model for Offer Management

After identifying valuable customers to prevent them from churn, a best offer selection mathematical model is proposed. The model calculates output for each customer an eligible offer. Furthermore, the model maximizes the total CLV by proposing the customers with the eligible offers. Base CLV is calculated by the Pareto/NBD model which is based on the RFM data.

The notation used in the mathematical model, sets, parameters, decision variables, and the proposed mathematical model are provided below:

Sets;

- $i =$ Customer index, $i \epsilon I$
- $j =$ Offer index, $j \epsilon J$

Parameters;

- $clv\_base_i =$ Base lifetime value of customer i. This amount should be interpreted as the expected present net worth of customer i for the following cases: (i) customer receives no offer or (ii) customer refuses the proposed offer. The base CLVs are estimated using Pareto/NBD model, Model 7 (see Table 4).
- $p_i =$ Churn probability of customer i to company before receiving the offer. Churn probabilities are estimated using the Pareto/NBD model, Model 7 (see Table 4).
- $o_i =$ Upper bound of discount for customer i,
- $w_i =$ Upper bound of offer j,
- $c_j =$ Discount rate of offer j,
- $f_i =$ Monthly payment of customer i to company before receiving the offer,
- $f_{avg} =$ Monthly average payment of customers,
- $\beta_{ij} =$ Probability of accepting offer j by customer i, $\beta_{ij} = p_i^{c_j \frac{f_{avg}}{f_i}}$,
- $\rho_{ij} =$ Churn probability of customer i after taking offer $\rho_{ij} = p_i \left(1 - p_i^{c_j \frac{f_{avg}}{f_i}}\right)$,
- $clv_{ij} =$ Modified Customer Lifetime Value (CLV New in Fig. 10) of customer i, upon accepting of offer j.
- $t =$ Monthly budget for discount,
- $M =$ Large positive number.

Decision Variables;

$$x_{ij} = \begin{cases} 1, & \text{if customer } i \text{ gets offer } j \\ 0, & \text{o/w} \end{cases},$$

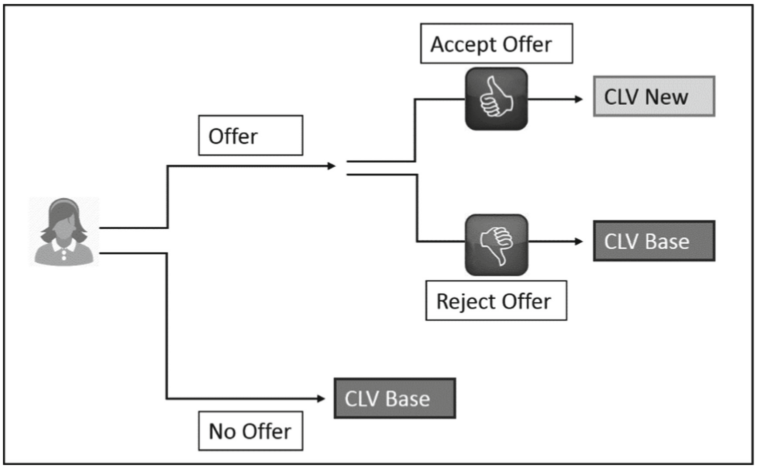


Fig. 10. Tree demonstrating the relation between (i) offer proposal (ii) offer acceptance/rejection (iii) CLV to be used for each case.

Mathematical model;

$$\max z = \sum_i \left[\sum_j x_{ij} \left[\beta_{ij} clv_{ij} + \left(1 - \beta_{ij}\right) clv\_base_i \right] + \left(1 - \sum_j x_{ij}\right) clv\_base_i \right] \tag{1}$$

$$\sum_i \sum_j f_i c_j x_{ij} \le t \tag{2}$$

$$\sum_i f_i c_j x_{ij} \le o_i, \quad \forall i \tag{3}$$

$$\sum_i x_{ij} \le w_j, \quad \forall j \tag{4}$$

$$x_{ij} \in \{0, 1\} \forall i, j \tag{5}$$

The objective function of the offer management mathematical model aims to maximize expected customer lifetime value while increasing retention of customers by giving the right offer. The expected CLV of each customer depends on (i) whether the customer receives an offer or not, and (ii) if the received offer is accepted or rejected by the customer. The expected CLV calculation idea used in the objective function is illustrated in Fig. 10. The constraints can be interpreted as follows: Constraint 2 states that offered discounts cannot exceed the total budget. Constraint 3 indicates the given offer cannot be greater than predetermined discount limit for each customer. Constraint 4 ensures that number of total proposed offers for each offer type cannot exceed the specified limit.

We made several assumptions in constructing links between offer acceptance/ rejection decision and its effect on updated CLV and churn probabilities. First of all it is assumed that the churn probabilities, p_i, are assumed to take values strictly greater than 0 and strictly less than one, in other words it is assumed that $p_i \in (0, 1)$.

This assumption is required as for the boundary values (that is $p_i = 0$ and $p_i = 1$) calculating p_{ij} values becomes problematic (see (7)). It is assumed that the tendency of accepting an offer can be estimated using (6). It is assumed that the probability of accepting an offer by a customer $\left(\beta_{ij}\right)$ is calculated depending on the customer's package refill payment (f_i), churn probability (p_i) and average package refill payment of all customers $\left(f_{avg}\right)$. Having lower churn probability, higher package refill payment from average and higher discount rate yield higher probability of accepting an offer, as shown in (6):

$$\beta_{ij} = p_i^{c_j \frac{f_{avg}}{f_i}} \forall i, j \tag{6}$$

In the same way, new churn probabilities $\left(\rho_{ij}\right)$ are calculated based on probabilities of accepting the offer $\left(\beta_{ij}\right)$ and churn probabilities (p_i). Having higher churn probability (p_i) and higher probability of accepting an offer $\left(\beta_{ij}\right)$ gives lower new churn probability after receiving an offer, as shown in (7):

$$\rho_{ij} = p_i \left(1 - p_i^{c_j \frac{f_{avg}}{f_i}}\right) \quad \forall i, j \tag{7}$$

New churn probabilities $\left(\rho_{ij}\right)$ are then used to calculate modified CLVs $\left(clv_{ij}\right)$. Modified CLV is the expected present net worth of customer i assuming customer accepts offer j as shown in (8):

$$clv_{ij} = \sum\nolimits_{n=1}^{n=\infty} \left(1 - p_{ij}\right)^n \left[\left(1 - c_j\right) clv\_base_i\right] \quad \forall i, j \tag{8}$$

In order to propose predetermined set of offers to a set of selected customers aforementioned mixed integer mathematical model is developed, and solved by using IBM CPLEX Solver.

We have solved problem with 6480 customers and 5 offers in Table 5. Offers are generated solely in the form of discounts.

Table 5. Offers.

Offer	Discount %
Offer 1	10
Offer 2	20
Offer 3	30
Offer 4	40
Offer 5	50

Different scenarios are run with the 6480 customers. Below are the details of the scenarios:

- Scenario1, the budget was set to 10.000 and offer count was set to 300. There is no cut off value for the customer churn probability.
- Scenario2, the budget was set to 10.000 and no offer limit was applied. There is no cut off value for the customer churn probability.
- Scenario3, the budget was set to 5.000 and offer count was set to 300. There is no cut off value for the customer churn probability.
- Scenario4, the budget was set to 5.000 and no offer limit was applied. There is no cut off value for the customer churn probability.
- Scenario5, the budget was set to 1.000 and offer count was set to 300. There is no cut off value for the customer churn probability.
- Scenario6, the budget was set to 1.000 and no offer limit was applied. There is no cut off value for the customer churn probability.

Scenarios are generated by changing overall budget limit and offer limits. Results for all six scenarios are presented in Tables 6, 7 and 8. Table 6 summarizes the number of customers who gets an offer. Table 7 presents the breakdown of the offer receiving customers over churn probabilities. For instance, in Scenario1, out of 741 offer receiving customers, 294 have churn probability less than 0.5 and the remaining 447 have churn probability greater than or equal to 0.5. Table 8 lists the distribution of offer suggestions over the possible list of offers for each scenario.

Table 6. Offer model scenario results.

	Budget	Offer limit	Cutoff	Offers received	% increase in CLV (compared to base CLV)
Scenario1	10.000	300	0	741	3%
Scenario2	10.000	inf	0	3.398	3%
Scenario3	5.000	300	0	532	3%
Scenario4	5.000	inf	0	1.498	5%
Scenario5	1.000	300	0	168	4%
Scenario6	1.000	inf	0	168	2%

Table 7. Churn rate breakdown.

	Churn rates		Total subs
	$p < 0.5$	$p >= 0.5$	
Scenario1	294	447	741
Scenario2	3.331	67	3.398
Scenario3	294	238	532
Scenario4	1.485	13	1.498
Scenario5	168		168
Scenario6	168		168

Table 8. Suggested offer distribution.

	Discount offers					
	Offer1 %10	Offer2 %20	Offer3 %30	Offer4 %40	Offer5 %50	Total subs
Scenario1	300	57	39	60	285	741
Scenario2	3.352	10	2	11	23	3.398
Scenario3	300	22	13	35	162	532
Scenario4	1.495	3				1.498
Scenario5	168					168
Scenario6	168					168
Scenario7	121	57	39	60	285	562

8 Conclusion

A significant percent of the mobile subscribers are still prepaid subscribers (see Fig. 6). In years, both the number of postpaid and prepaid subscriber has converged to each other as a result of increased data usage due to 4G network upgrade by the mobile operators. In this study, we concentrated on developing an offer management framework, which includes a state of the art mathematical model that is based on important estimated parameters for customer relations management, namely churn probabilities and CLVs. It is seen that, instead of proposing an offer to every churner, the mathematical model gives decision makers the flexibility of addressing offers to the most valuable subscribers who have higher potential to accept offers.

CLV estimation is seen as an important part of the offer management framework, and without CLV integration, the mobile operators should accept the burden of high rate of false positives in their offer proposals. Another important finding is, in estimating CLVs, segment and tenure information turns out critical. It can be seen from the (Fig. 9a–f) that Pareto/NBD model yields good results for the youth and mass segments in the test period only when tenure information is incorporated. It is also found that for the non-segmented customers (other segment) the Pareto/NBD model fails, which seconds the claim that both segment and tenure play critical role in order to obtain reasonable predictions from Pareto/NBD model.

In general, it should be said that there is still room for improving the parameter estimation for Pareto/NBD model. There are some different studies about the parameter estimation [13–15], which may be followed to come up with a better estimation technique.

Lastly, it is planned to conduct an A/B testing in a real world setting, where responses of the subscribers will also be collected and the actual performance of the proposed framework will be revealed.

References

1. Kumar, V., Rajan, B.: Profitable customer management: measuring and maximizing customer lifetime value. Manag. Acc. Q. **10**(3), 1 (2009). Spring 2009
2. Can, Z., Albey, E.: Churn prediction for mobile prepaid subscribers. In: Proceedings of the 6th International Conference on Data Science, Technology and Applications Volume 1, DATA 2017, ISBN 978-989-758-255-4, pp. 67–74 (2017)
3. Pfeifer, P., Haskins, M., Conroy, R.: Customer lifetime value, customer profitability, and the treatment of acquisition spending. J. Manag. Issues **17**(Spring), 11–25 (2005)
4. Hwang, H.: A dynamic model for valuing customers: a case study. Adv. Sci. Technol. Lett. **120**, 56–61 (2015)
5. Fader, P., Bruce, H., Ka, L.: RFM and CLV: using iso-value curves for customer base analysis. J. Mark. Res. **42**, 415–430 (2005)
6. Wei, J., Lin, S., Wu, H.: A review of the application of RFM model. Afr. J. Bus. Manag. **4**(19), 4199–4206 (2010). December Special Review
7. Zabkowski, T.: RFM approach for telecom insolvency modeling. Kybernetes **45**(5), 815–827 (2016)
8. Huang, B., Kechadi, M., Buckley, B.: Customer churn prediction in telecommunications. Expert Syst. Appl. **39**(1), 1414–1425 (2012)
9. Segarra-Moliner, J., Moliner-Tena, M.: Customer equity and CLV in Spanish telecommunication services. J. Bus. Res. **69**(10), 4694–4705 (2016)
10. Bahnsen, A., Aouada, D., Ottersten, B.: A novel cost-sensitive framework for customer churn predictive modeling. Decis. Anal. **2**, 5 (2015)
11. Khajvand, M., Zolfaghar, K., Ashoori, S., Alizadeh, S.: Estimating customer lifetime value based on RFM analysis of customer purchase behavior: case study. Procedia Comput. Sci. **3**, 57–63 (2011)
12. Platzer, M.: Stochastic models of noncontractual consumer relationships. Master thesis at the Vienna University of Economics and Business Adminstration under the Supervision of Dr. Thomas Reutterer (2008)
13. Homburg, C., Koschate, N., Hoyer, W.D.: Do satisfied customers really pay more? A study of the relationship between customer satisfaction and willingness to pay. J. Mark. **2**(69), 84–86 (2005)
14. Ma, S., Liu, J.: The MCMC approach for solving the Pareto/NBD model and possible extensions. In: Third International Conference on Natural Computation, ICNC 2007, vol. 2. IEEE (2007)
15. Abe, M.: Counting your customers one by one: a hierarchical Bayes extension to the Pareto/NBD model. Mark. Sci. **28**(3), 541–553 (2009)
16. Platzer, M., Reutterer, T.: Ticking away the moments: timing regularity helps to better predict customer activity. Mark. Sci. **35**(5), 779–799 (2016)

Construction of Semantic Data Models

Martha O. Perez-Arriaga(✉), Trilce Estrada, and Soraya Abad-Mota

University of New Mexico, Albuquerque, NM 87131, USA
{marperez,estrada,soraya}@cs.unm.edu

Abstract. The production of scientific publications has increased 8–9% each year during the previous six decades [1]. In order to conduct state-of-the-art research, scientists and scholars have to dig relevant information out of a large volume of documents. Additional challenges to analyze scientific documents include the variability of publishing standards, formats, and domains. Novel methods are needed to analyze and find concrete information in publications rapidly. In this work, we present a conceptual design to systematically build semantic data models using relevant elements including context, metadata, and tables that appear in publications from any domain. To enrich the models, as well as to provide semantic interoperability among documents, we use general-purpose ontologies and a vocabulary to organize their information. The resulting models allow us to synthesize, explore, and exploit information promptly.

Keywords: Table understanding · Information modeling
Semantic data model · Semantic interoperability
Information extraction

1 Analyzing Scientific Publications

Modern research relies significantly on exploring and building on top of existing scientific production. But in recent decades this production has increased exponentially. With this growth, new challenges have to be addressed. Exploration of very large digital corpora can be done manually, but it is time consuming, tedious, and potentially results in incomplete analyses. On its automatic form, this exploration involves two main phases: document retrieval and analysis. Document retrieval can be done efficiently through keyword search, phrase matching, topic categorization, and other more sophisticated information retrieval techniques. The analysis phase has to consider the semantics of the document, as well as its qualitative and quantitative content. However, this information can be hard to get, and in many cases it is buried in tables whose relationships have to be inferred.

Because of the exposed problems, we develop a conceptual design to extract semantic information from digital documents. In general, the design of a data model requires the knowledge of entities that interact with each other. Our design allows us to systematically compose a data model without knowing its elements

© Springer International Publishing AG, part of Springer Nature 2018
J. Filipe et al. (Eds.): DATA 2017, CCIS 814, pp. 46–66, 2018.
https://doi.org/10.1007/978-3-319-94809-6_3

a priori. Thus, we recognize its components and interactions at the same time of analyzing a document.

According to Peckham and Maryanski [2], a *semantic data model* includes two main components: (1) relationships between entities and (2) well-defined semantics for the relationships. The richness of information in a digital publication provides the elements to create a semantic data model. The concepts in a publication are entities that appear along the narrative of a study, in a publication's context, and embedded in tables or other structures in a document. To define, disambiguate, and enrich entities, additional information can be extracted from external sources of knowledge, such as ontologies and the Internet, to use them as semantic annotations. A *semantic annotation* is an association between a concept in a document and a definition contained in an established database or ontology.

To enrich concepts within publications, several works integrate semantic annotations in these documents [3,4]. For instance, BioC [4] provides a design to define annotations from publications of the biomedical field, offering tools for developers to create definitions in XML. The tools are simple to use, however, it is necessary to have programming skills to take full advantage of them. The International Association of Scientific, Technical, and Medical publishers points out the importance of publishing with semantic practices [5]. To include semantic annotations before publishing documents, the Semantic Web Science Association [6] defines rules to improve promoting and sharing articles related to the Semantic Web. Peroni [7] presents semantic publishing and referencing ontologies to create metadata and include semantic annotations in scientific articles. However, these publishing practices are far from being extensively adopted and the bulk of scientific publications in most fields lack even minimum annotation standards. Searching for concepts in documents lacking semantic annotations makes the extraction process harder. Thus, an ideal process to analyze information should be interoperable and include a direct way to identify elements of interest, such as concepts with semantic annotations and context. Semantic interoperability refers to integrated information systems that are able to hide syntax and structural heterogeneity from data sources [8], while providing shared and unambiguous information.

Our goal is to automatically build semantic models on the fly to characterize and annotate documents. Our model creation is fully automatic, as it does not need prior information regarding concepts, entities, or metadata standards. It is also interoperable, as our conceptual design provides an extensible and standardized mechanism for unambiguous information exchange through semantic annotations and provenance. Finally, it is exhaustive, as it takes advantage of qualitative information, found through the document's narrative, as well as quantitative information, found in implicit relationships in tables.

To build data models, we (a) identify and extract context, metadata, and concepts from a publication and its tables; (b) detect and extract semantic relationships; and (c) characterize a semantic data model from each publication. A semantic data model derived from our conceptual design provides a semantic characterization of quantitative and qualitative relationships in the document. Our contributions in this paper are twofold: a formal definition to systematically generate semantic data models to facilitate synthesis, extraction, and comparison of specific information; and a characterization of data models from scientific documents of any domain for semantic interoperability. We extend our previous work [9] introducing the conceptual design of semantic data models for synthesizing publications. Although this design is for publications, the analysis of large volume of information in any domain can benefit using a conceptual design. Using a working example, we demonstrate the possibilities of finding semantic relationships at each level of a model, which can potentially guide building knowledge bases. Furthermore, we improve the methods to disambiguate entities and extract semantic relationships, deepen on the details of our characterization that encloses data integration and semantic interoperability, and measure similarity of semantic relationships to compare information among data models.

In the remainder of this work, Sect. 2 presents a review of related work. Section 3 presents a framework to develop the models, including the organization of a semantic data model using a working example. Section 5 presents an assessment of the methods to generate our models. Section 6 presents the conclusion and future work to be undertaken.

2 Related Work

We review methods used to obtain the main elements to compose the data models. In particular, we present related work to identify and extract semantic relationships, and to summarize information from digital documents.

Important work to extract semantic relationships include the Never Ending Learning Language (NELL) [10], PATTY [11], and Open Information Extraction (OIE) methods [12–14]. The NELL [10] approach creates a knowledge base of categories and relationships from the Web. This approach extracts patterns and relationships, classifies noun phrases, and infers new relationships from their knowledge base. On the other hand, the approach PATTY [11] is based on frequent itemset mining. PATTY also detects relationships from the Web, but this approach uses syntactic, ontological, and lexical features. The Open Information Extraction approach finds new relationships without using patterns [12,13]. Fader et al. state *"OIE can find incoherent and uninformative extractions."* and improve it developing Reverb [15]. Reverb finds relationships using part of speech tags, noun phrase sentences, and syntactic and lexical constrains. Reverb uses a corpus built offline with Web sentences to match arguments in a sentence heuristically, and finds a confidence percent for each relationship using a logistic regression classifier. Reverb and R2A2 are part of the second generation of OIE [14]. R2A2 includes an argument learner, and linguistic and statistical analyses to extract relationships. The learner consists of classifiers using specific patterns, punctuation, and part of speech tags to identify arguments.

We used Reverb in our previous work [9], which detected semantic relationships in publications with a confidence measure. The high ranked relationships found by Reverb missed relationships containing entities derived from tables. Therefore, we used the wealth of information in tabular structures and a context to retrieve relationships with relevant arguments from a publication. Similarly to OIE methods, our approach uses an unsupervised learning and heuristics. However, we focus our attention on finding relationships with entities of interest. Methods for extraction of semantic relationships, which are based on Web and text formats, lack methods to preserve the association between a source document and its relationships, and lack an organization of relationships per document. To overcome these issues, our framework integrates relevant information from publications as relationships and enhances them with external sources of information, organizes the relationships in a common characterization document, preserving the original publication's provenance for further consultation.

The second part of this section reviews work to represent semantic information and to summarize information from digital publications. Hull and King [16] describe semantic data models and their ability to (1) separate concepts in physical and logical forms, (2) minimize overload of relationships, and (3) represent abstract concepts and relationships. The models have been used to include semantics in a database schema, which is difficult to represent and implement. Peckham and Maryanski [2] analyze semantic data models and determine the two main components: relationships between entities and well-defined semantics for the relationships. The models represent data objects and relationships known *a priori*. For example, the *World Traveler Database* modeling [16], which represents relationships among known entities. Conversely, our conceptual design uses entities from a context and tabular layouts embedded in any digital document, as well as relationships from internal elements in a document (i.e., tables and text), and external sources of information, such as the general ontology DBpedia and the Internet. To disambiguate entities, we use DBpedia [17] to describe each entity. DBpedia is a curated structured ontology with entities' properties. Furthermore, DBpedia contains information that follows the Semantic Web principles. If DBpedia lacks an entity's information, this work uses the unsupervised approach *Latent Semantic Indexing* [18] and a publication's context to disambiguate an entity, providing an explanation of the entity at hand from the Internet. Furthermore, we use the Semanticscience Integrated Ontology [19] to provide structure and semantics to the relationships derived from scientific publications. Finally, to represent a publication's metadata, we use the general vocabulary schema.org [20] in a machine-readable format.

Summarization provides a short representation of the content of publications and can be used to analyze them faster. Automatic approaches advance this research area. Nenkova et al. [21] present an extensive description of methods for summarization in different areas, such as medical, journals, and news. Teufel and Moens [22] point out differences in summarizing documents from science or news, the latter being more repetitive and less concise in their arguments. We also notice that scientific documents often have space restrictions and their narrative is succinct. Allahyari and colleagues [23] present a survey of text summarization methods and point out the importance of ontologies in summarization. Several approaches to summarize publications use rethoric analysis using the sentences in each section. Teufel and Moens analyze relevance in sentences and rhetorical status, based on analyzing organization, contribution, and citations in a scientific article. Baralis et al. [24] use frequent itemsets to summarize documents. Our framework generates a short summary, synthesizing a publication based on ontologies, relevant concepts, and semantic relationships.

3 Generation of Semantic Data Models

To find concrete information embedded in a vast collection of scientific documents, we develop a framework to create semantic data models. Figure 1 depicts this process. First, the framework ingests a document in PDF, XML, or text formats. Then it extracts metadata and context from the document's title, keywords, and authors. Then it parses tabular information to perform discovery and interpretation of tabular information. This information is then used to guide the process of entity and relationships discovery. Finally, all of these pieces, metadata, context, entities and semantic relationships, are used to form the semantic data models that characterize the document. These models can be further used for summarization, annotation, and searching purposes. In the following sections, we present the approach to recognize the different elements in a digital publication and compose our models.

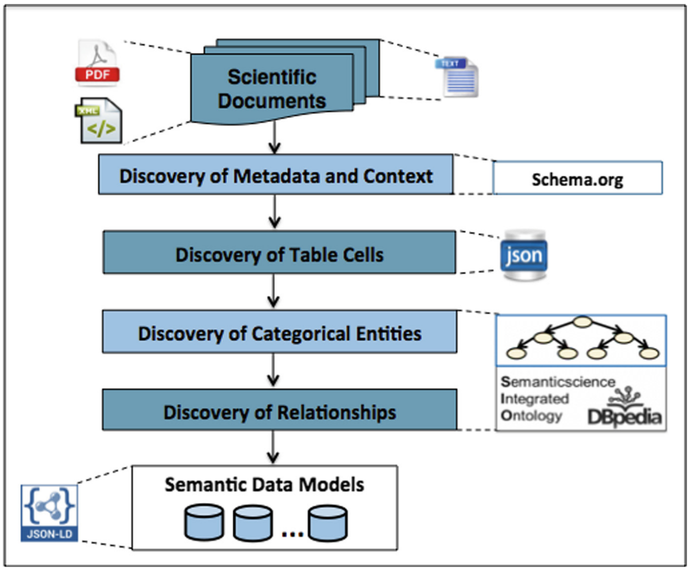

PDF
XML
Scientific Documents
TEXT
Discovery of Metadata and Context
Schema.org
Discovery of Table Cells
json
Discovery of Categorical Entities
Discovery of Relationships
Semanticscience Integrated Ontology
DBpedia
JSON-LD
Semantic Data Models
...

Fig. 1. Framework to generate semantic data models.

3.1 Discovery of Metadata and Context

Metadata. Represents the key information to identify a publication, such as title and author. It is important for digital identification, resource discovery, electronic resources organization, interoperability, archive and preservation [25]. Metadata can be represented in different formats, for instance, using the standard generalized mark-up language, extended markup language (XML), or even using a database to represent and manage metadata attributes [26]. To recognize metadata in digital publications, we directly identify tags associated with metadata terms. Our method searches for tags describing the metadata elements in documents, including `<author>`, `<article-title>` and `<keywords>`. If tags are not found, our framework accepts a digital document in plain text or PDF format to search for metadata. A digital document in PDF is converted to XML using PDFMiner [27]. Then, we use pattern matching on the first page of the publication to extract this information. The keywords defined in a publication are used to represent its context. If keywords are not defined in a document, we search for them. For representing the extracted metadata, we use the schema.org vocabulary [20] because it can represent information from different domains with minimum dependency on ontologies. In particular, we use properties such as creator, headline, and keywords of the scholarly article object to identify the different metadata elements. By using a controlled vocabulary, we ensure an homogeneous data representation, that is vital to facilitate interoperability, sharing, and data collaboration.

Context. Represents the conceptual framework surrounding a publication. Current publishing standards dictate that publications need to define keywords to describe the domain and sub domains of the document. Additionally, the text itself contains concepts that can be used as context as well. To extract the publication's context, our framework first searches for keywords. If keywords are not defined, then the text undergoes a preprocessing step, where we eliminate stop words, and small and large length words. Each word is then used as a unigram. We use the Term Frequency Inverse Document Frequency (TFIDF) method to score the relevance of each word in the context of the document as opposed to a word that is frequently used regardless of its context. This method uses term frequency (TF) to detect a word's relevance to a particular document, and it uses the inverse document frequency (IDF), which is the logarithm of the ratio obtained using total number of documents in a collection and the number of documents where the term appears. To calculate IDF we use Wikipedia as a wide collection of documents, which contains more than five million of documents to date [28]. Once we calculate the TFIDF score for every word in the document, we use the five terms with highest scores as additional keywords of the publication. These keywords become part of metadata that characterizes the semantic model of the publication.

3.2 Discovery of Categorical Entities

In addition to context, we extract categorical entities from keywords, text, and tables. In this section we describe how to recognize, disambiguate, and annotate entities as presented in [9], as well as how to find and associate annotations to entities.

Information Extraction from Tables. For entity extraction, our first step is to explore tables in search for quantitative information. Tables present concise associations in a simple representation. A study found that 74.6% of tables appear in the results section in scientific publications of different domains [29]. The main challenge of discovering and extracting tables from documents is that they are embedded within text and other elements, such as equations and graphics. Also, they are created using different formatting and layouts.

Our framework is able to discover table cells from XML and PDF documents. For the former, we extract tables content from well-formed XML documents using the tags `<table-wrap>` and `<table>`. We use Xpath [30] to detect tags and find a table within a document. Then, the table cells are organized by rows in JavaScript Object Notation (JSON) format. In addition, we determine if a cell is a header or data using the tags `<thead>` and `<tbody>`. A header is a label that represents and groups table information, rows or columns, while data is the actual content of a cell that compose the body of a table. Finally, we detect data types for each table cell using regular expressions. The process for table discovery in PDF documents is more elaborated. We use our previous work in TAO [31] that consists of three modules: (1) Document conversion, which uses PDFMiner [27] to convert a PDF document to an extensive document in XML. The output includes separate XML tags for every text box, character, and space, including its coordinates, font, and size. In this format it is not possible to identify a table and its content without further processing (2) Table detection, parses the output in XML using a combination of layout heuristics to detect table candidates within the document. (3) Table extraction uses table candidates and supervised learning to find a table's content. Specifically, we use k-nearest neighbor logistic regression to find alignment of columns, which determine an actual table. Then, we extract the content of each table cell and analyze cells to classify their data type and function, that is, header and data. TAO's output is saved in JSON format as well.

Entity Recognition and Annotation. To recognize entities, each table cell's content whose data type is a string, undergoes natural language processing using TextBlob [32], which executes a noun phrase analysis to discover entities. To make sure that every entity found is unambiguous we identify it and annotate it through DBpedia [17]. To this end, we search the entity in DBpedia using its naming convention, which utilizes its first capital letter and concatenates words with an underscore. For example, the entity *diabetes management* converts to "Diabetes_management." DBpedia redirects searches that refer to the same concept automatically. Searching for "Glycemic_control" or "Diabetes_treatment", DBpedia returns the entity's resource for "Diabetes_management" because the former concepts appear in the property `wikiPageRedirects` of this entity.

DBpedia offers a Web page, that is, a resource for each entity containing a description, a type, and properties of a particular entity. For instance, the entity *Diabetes* has the resource http://dbpedia.org/page/Diabetes_mellitus, which defines this entity with type *disease* and includes several properties. In particular, the property *abstract* comprises a summarized description of the entity, and we use it as an annotation *"Is A"* for the entity. We also keep the URI and store it as an annotation to identify the entity and for further consultation.

Entity Disambiguation. Even though DBpedia is a curated structured ontology, it does not contain a description for every entity possible. Moreover, if a concept has more than one meaning (i.e., it is ambiguous), DBpedia shows a list of possible concepts under the property *wikiPageDisambiguates*. For example, currently DBpedia shows forty different concepts for the entity *Race*. When the property *wikiPageDisambiguates* exists, we use a global search on the Internet to disambiguate and find a URL to unambiguously explain an entity. To narrow down the search results and to obtain more accurate results, we augment the search with the document's context.

We use the Web Search API from Microsoft Cognitive Services [33] and a tailored Latent Semantic Indexing (LSI) analysis [18] to find the closest explanation to an entity and its context. We use LSI because it performs well categorizing documents with only several hundred dimensions [34]. In particular, we search for an ambiguous entity x and retrieve at most n documents $D = d_1, d_2, \ldots, d_n$ containing x, where $n = 50$. For each $d_i \in D$, we normalize a vector with at most 300 relevant words. Then, we create a normalized query vector q containing the context, title, and abstract of a publication, as well as sentences containing the entity of interest in the original publication. After applying LSI, we select the most similar vector $d_i \in D$ to vector q and we recover its associated URL to use as an identifier for entity x. The URL is a non-formal annotation to represent entity x. It is non-formal because differently from a URI, a URL may change over time, and because its content is not very often curated.

3.3 Discovery of Relationships

Through the process described above, we create a list of unambiguous and annotated categorical entities per document. We then used these entities to find meaningful semantic relationships in the publication. We base our definition of a semantic relationship on concepts by Dahchour et al. [35], who defines a *semantic relationship* as a binary relationship that is domain independent and represents static constrains and rules, such as, classification, generalization, grouping, and aggregation. We discover relationships using table cells and text. To find relationships from tables, we relate each data table cell with its header. The table cells' organization containing rows, column numbers, content, and cell function facilitate this process. We also use the text of the document and table's caption to find additional relationships.

Our first approach, described in [9], used the open information extraction method Reverb [15] to identify relationships. However, Reverb was not able to identify many relevant relationships associated with entities derived from tables. Therefore, we designed a more extensive relationship identification process comprising four steps: (a) segmentation, (b) pattern matching, (c) part of speech tagging, and (d) relationship composition. First, we recover the text of a publication and perform segmentation of sentences, using the Python Natural Language ToolKit [36]. Once we recover all sentences of each publication, the categorical entities are used to search for pattern matching in these sentences. If a sentence contains an entity, we apply part of speech tagging. Then, we detect combinations of tags with verbs, such as `VB`, `VBZ`, `MD VB`, `MD have VBN`, `MD has VBN`, `MD VBZ`, and `VB VBZ`. The verb tags assist to identify sentences containing an action between an entity and other text. If a verb combination is found, we use it as a relationship. For example, the sentence *obesity is related to lack of exercise* contains the relationship *is related*. Last, we use the Semanticscience Integration Ontology [19] (SIO) to formally represent a relationship with its ontology definition. We use pattern matching to relate a definition of a relationship from SIO with our extracted relationships. If a match is not found, we keep a verb as the relationship itself. For instance, the sentence *the experiment is derived from our study* has the verb *is derived*, which is denoted in SIO by the label *is derived from* and id *SIO_000244*. But the sentence *obesity increases high blood pressure* has the verb *increases*, which is not included in SIO, hence, *increase* represents this relationship.

3.4 Organization of a Semantic Data Model

Our proposed semantic data models integrate important information derived from elements found in publications regardless of their domain. Specifically, to compose a semantic data model, we use associations found in tabular patterns and unstructured text. Thus, we define a semantic data model as the structured representation of semantic elements in a publication. Specifically, it contains metadata and context, entities, and semantic relationships.

A semantic model generated by our framework is represented and stored as a JSON-Linked Data (JSON-LD) object. JSON-LD [37] was created to facilitate the use of linked data. A document in JSON-LD can define a type of the information represented using a context and its relationships are not limited by a specific cardinality. This format is compatible with RDF and other variants such as N-quads [38].

Our framework uses JSON-LD to generate a descriptive document with the publication's metadata, entities with annotations, and the actual information from extracted relationships (i.e., relationship identifier, arguments, definition of relationship, and definition identifier). The document in turn contains other links, such as the ones used to annotate entities and to define relationships. In addition, it contains a model context, which denotes the environment of the components of the semantic data model. Note that the context of a semantic data model is different to the context of a publication, which is represented using keywords. Using as a working example the publication: *Neuropeptidomic Components Generated by Proteomic Functions in Secretory Vesicles for Cell-Cell Communication* [39] by Vivian Hook et al., we find the entity *Neuropeptides.* We represent it in JSON-LD, which includes a model context indicating that this entity is identified by a resource from the ontology DBpedia.

```
{
 "context": "{http://dbpedia.org/page/}"
 "Neuropeptides": {
       "id":"Neuropeptides",
       "abstract":"small protein-like molecule (peptides) used
                   by neurons to communicate with each other"
}
```

To compose the semantic data model to be used as a synthesis of a document, we build a semantic hierarchy of its different components. Where the top components represent the most general information and the bottom layers represent specific identifiers and annotations. Figure 2 shows a partial graphic depiction of this hierarchy.

The first layer of the model corresponds to the *metadata and context*, which includes identifiers for keywords, headings, and provenance. The next layer contains *entities*, where specific concepts are identified as relevant entities in the document (e.g., Neuropeptide, pain). Then the following layer contains *semantic relationships*, which link one or more entities and resources (e.g., Has Attribute, Is A). Finally, the bottom layer contains specific *annotations* for both entities and relationships The discovery of relationships in our semantic models allows researchers to find results, experimental settings, and possible associations among concepts in scientific documents. The hierarchical organization of our data model facilitates finding relationships among entities, and later among publications.

4 Conceptual Design of Semantic Data Models

To provide semantic interoperability among digital publications, we present a conceptual design to create semantic data models systematically. From the elements contained in a publication, we identify the finite components of a *semantic data model* as a 3-tuple $SDM = \{M, E, SR\}$. M represents metadata and context, E entities, and SR semantic relationships. Fundamental elements subsumed in this model are table cells that serve to find entities E and semantic relationships SR. To depict the conceptual design of these components, we use the Entity-Relationship (ER) model as described by Elmasri and Navathe [40]. Rectangles indicate entities, diamonds indicate relationships, and values at both ends of a relationship (min, max) are a combined cardinality/participation notation, which indicate the structural constraint of the minimum and maximum participation of an entity in a relationship.

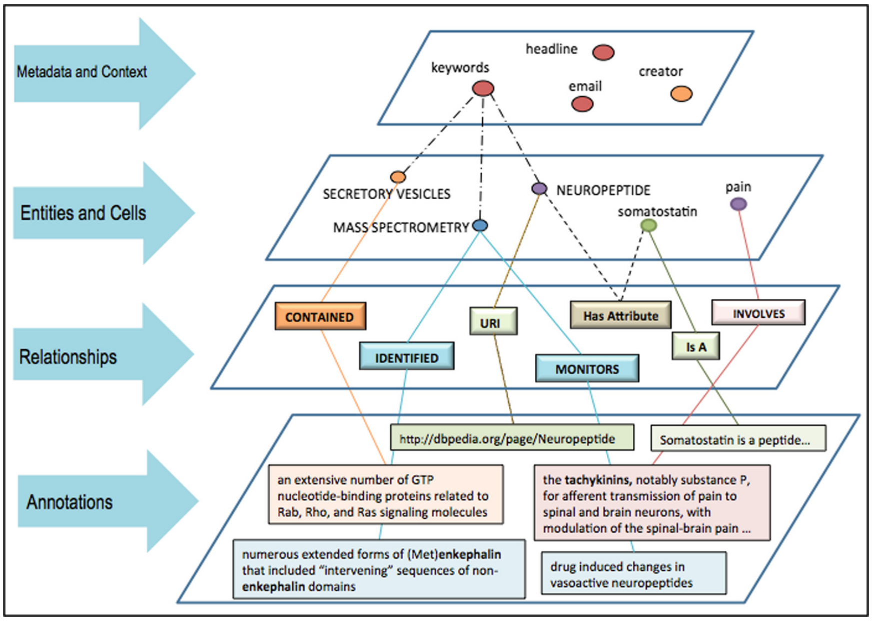

Metadata and Context
Entities and Cells
Relationships
Annotations
headline
keywords
creator
email
SECRETORY VESICLES
NEUROPEPTIDE
pain
somatostatin
MASS SPECTROMETRY
CONTAINED
URI
Has Attribute
INVOLVES
IDENTIFIED
MONITORS
Is A
http://dbpedia.org/page/Neuropeptide
Somatostatin is a peptide...
an extensive number of GTP nucleotide-binding proteins related to Rab, Rho, and Ras signaling molecules
the tachykinins, notably substance P, for afferent transmission of pain to spinal and brain neurons, with modulation of the spinal-brain pain ...
numerous extended forms of (Met)enkephalin that included "intervening" sequences of non-enkephalin domains
drug induced changes in vasoactive neuropeptides

Fig. 2. Semantic data model from working example.

Metadata and Context. The first component of our model, M, includes metadata and context of a publication. Figure 3 shows the conceptual design of this component, which includes a context of a publication as a set of keywords, a title of a publication, author(s), and corresponding email. A data model representing a publication is primarily identified with a unique title, which can also relate to a unique Digital Object Identifier. A publication is associated to one or more authors. If a first author has collaborator(s), then for each pair *first author-collaborator* an *isCoauthor* relationship exists. Finally, each author has one email address.

Cells and Categorical Entities. Figure 4 shows the conceptual design of cells C and categorical entities E. First, the element C is composed by a set of cells contained in a given table. The definition of a semantic relationship between cells is indicated by a **header cell**, which *hasAttribute* contained in a **data cell**. A header cell can be related to more than one data cell, while a data cell has to have exactly one primary header cell.

The element E represents a set of entities. The entities can be found in tabular structures and within the context of a publication. To detect entities E, we can use cells in C with data type string and keywords representing a context. Figure 4 shows the semantic relationship **cell** *canBe* an **entity**, to indicate that an entity can be found in a cell. The relationship **entity** *composedBy* **keyword** indicates that E at least contains one keyword, and that a keyword can be in one or more entities.

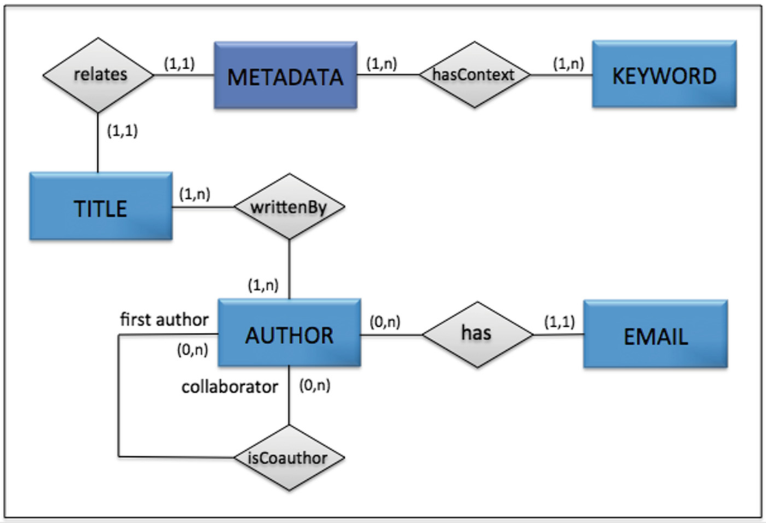


Fig. 3. Metadata components. Reprinted from [41]

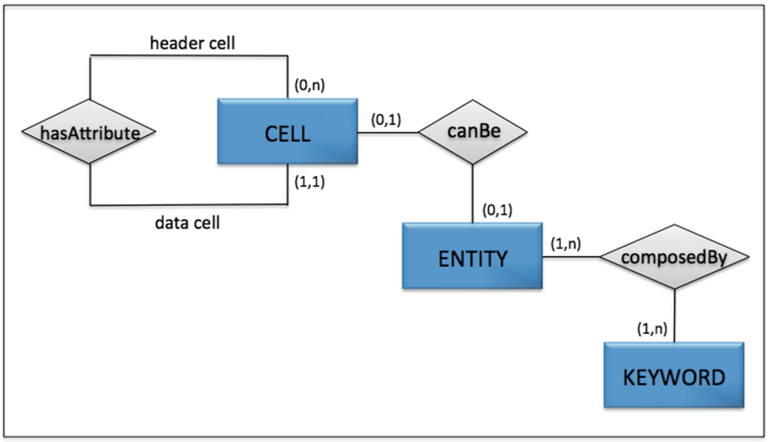


Fig. 4. Cells and entities. Reprinted from [41]

Semantic Relationships. The last element of our model contains semantic relationships SR. The components of SR include entities, text, and semantic annotations. The annotations for categorical entities contain information from DBpedia or the Internet. Figure 5 shows the relationship of an entity and its semantic annotations. In particular, a semantic annotation consists of two relationships: **ontology** *defines* an **entity** and a **URI** *describes* an **entity**. A URI is a Universal Resource Identifier and a URL is a Uniform Resource Locator. The relationships for a semantic annotation include *defines* or *IsA*, and *describes*.

If semantic annotations are not found, an entity can relate to a URL, where a **URL** *explains* an **entity**. The relationship *explains* is not necessarily a semantic annotation because a URL can change over time and may not contain a formal definition. Figure 5 also shows **entity** *associates* to a phrase in **text**.

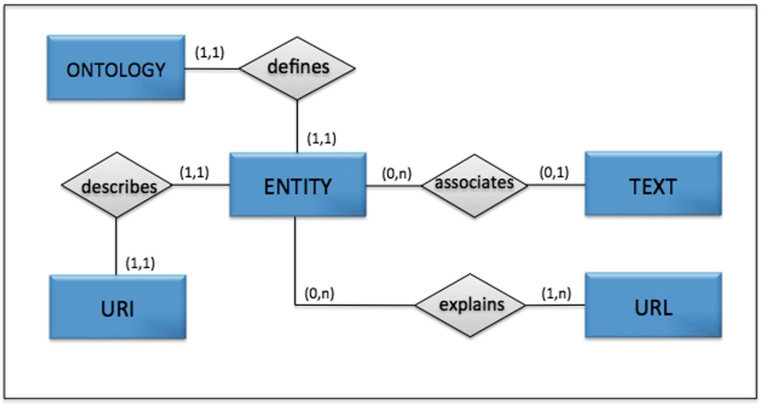


Fig. 5. Semantic relationships. Based on [41]

The semantic relationship *associates* can acquire different labels describing relationships from text. We can find these labels defined in the Semanticscience Integrated Ontology [19], which contains formal definitions of relationships, for instance, *is derived from* and *has basis*. The relationships from SIO are not exhaustive, therefore, verbs can also represent relationships, as explained in Sect. 3.3.

5 Evaluation

To assess the effectiveness of our method, we extend the evaluation from [9] for discovery of semantic relationships. In particular, we use two datasets. The datasets contain publications downloaded from the PubMed Web site ftp://ftp.ncbi.nlm.nih.gov/pub/pmc/oa_bulk/. The first dataset $PubMed_1$ comprises fifty publications with various topics, 449 text pages and 133 tables with different tabular formats embedded either in one- or two-column documents. We prepared a gold standard with manually-defined entities for each table in the publications. With similar layout characteristics, the second dataset $PubMed_2$ is composed by seventy five publications with 670 text pages and 190 tables.

We evaluated our methods quantitatively and qualitatively with four experiments to measure our framework's ability to (1) recognize and annotate entities, (2) disambiguate entities, (3) identify semantic relationships between entities, and (4) generate similar semantic relationships among data models. The dataset $PubMed_1$ is used in the first three experiments.

5.1 Experiment: Entity Recognition and Annotation

The first experiment measures our method's ability to recognize and annotate entities. We use *recall* and *precision* to obtain the F1-measure for entity recognition. *Recall* is the ratio between the correct number of entities detected and the total number of entities in a table. *Precision* is the ratio between the correct number of entities detected and the total number of entities detected. *Recall* and *precision* are also used to evaluate accuracy for entity annotation.

Our gold standard contains 2, 314 entities, and our method recognized 1, 834. We obtained a recall of 79.2% and a precision of 94.3%, yielding a F1-measure of 86.1% (see Table 1).

For entities annotated, we found 1, 262, from these entities, 72.5% were found in DBpedia. However, only 785 contained a unique description in DBpedia, that is, 45.1%. For the rest, our method used the context of a publication and the LSI process described in [9] to annotate 955 entities (i.e., we annotated 54.9% of entities with our LSI + context method). From the 1, 834 entities recognized, 1, 740 were correctly annotated, yielding a recall of 94.8%, precision of 97%, and F1-measure of 95.9%.

Table 1. Experiment entity recognition and annotation. Reprinted from [9].

Entity recognition			
Entities	Recall	Precision	F1 measure
Recognized	0.79	0.94	0.86
Annotated	0.95	0.97	0.96

5.2 Experiment: Entity Disambiguation

The second experiment evaluates our entity disambiguation methods. We quantify how including the context of each publication affects the entity disambiguation process. From 1, 740 entities, 955 of them needed disambiguation. The first part of this experiment did not use a context to disambiguate entities. Still, our framework was able disambiguate 838 entities correctly without context, with precision 89%, recall 87%, and an F1-measure of 88%. During the second part of this experiment, we used keywords as a context, yielding 900 entities disambiguated with precision of 95%, recall 94%, and an F1-measure of 94.5%. Table 2 reports these results.

To evaluate the quality of the disambiguation process, the URLs obtained from this process were manually classified as reliable and non-reliable (see Table 2 column *Non Rel. URLs*). Reliable URLs were derived from known organizations including universities, digital libraries, and hospitals. While non-reliable URLs required further analysis. Manually reviewing URLs ensures that a Web page or document is related to an unresolved entity and its publication. However, it does not ensure that the explanation of the entity is correct. The non-reliable URLs when no context was used, were 117, that is 12.3% of the total disambiguated entities. The non-reliable URLs found when additional context was used, were 55, that is 5.8%. Therefore, the use of context reduced the non-reliable links by more than half.

Table 2. Experiment entity disambiguation. Reprinted from [9].

Entity disambiguation				
Method	Recall	Precision	F1 measure	Non Rel. URLs
No context	0.87	0.89	0.88	12.3%
Context	0.94	0.95	0.94	5.8%

5.3 Experiment: Identification of Semantic Relationships

The third experiment evaluated both quantitatively and qualitatively the relationships found by our methods. The relationships extracted from tables and text contain relevant concepts in a structured presentation. First, we measure the total number of relationships with high rank, in particular a confidence score ≥ 0.70 using Reverb. Second, we measure the total number of relationships extracted with our new method using an unsupervised method and relevant entities. Furthermore, we manually evaluated qualitatively the relationships classifying them as complete and incomplete. The latter refers to a relationship missing an argument.

From tables in $PubMed_1$ we found $11,268$ relationships. Exclusively from text while using Reverb, we found 865 high ranking (confidence ≥ 0.70) relationships. On the other hand, using categorical entities with our new approach, we found $1,397$ relationships. A human judge analyzed the completeness of relationships manually. Results are reported in Table 3. From the total number of relationships obtained from tables, $10,102$ or 89% of relationships were complete. From the relationships extracted from text using Reverb, 703 or 81% of relationships were complete. The rest, that is 19% was labeled as incomplete. For the approach using entities from tables and text, a judge identified $1,336$ or 90% of them as complete, while the rest 10% was labeled as incomplete. Our improved approach found almost twice the number of relationships than the number found by Reverb. Thus, the we were able to empirically demonstrate that the number of extracted relationships increased using relevant entities.

Table 3. Experiment semantic relationships. Based on [41]

Semantic relationships			
Method	Rel. found	Rel. complete	% complete
Tables	11,268	10,102	89%
Text reverb	865	703	81%
Text entity	1,397	1,336	90%

Using a publication's context (i.e., keywords) and concepts from tables' cells to find entities ensures their relevance. But we still found several incomplete or malformed relationships from text containing information from tables. An area of improvement for our approach is to extract and eliminate irrelevant tables' information in a document to eliminate some false positive relationships from text.

5.4 Experiment: Determining Semantic Similarity

The last experiment used $PubMed_2$ to generate 75 semantic data models. The sets of relationships contained in the models were compared with one another using cosine similarity. Figure 6 shows the matrix of scaled similarity among data models. Every cell i, j in the matrix represents the similarity of model i with model j, thus, the matrix is symmetrical. A lighter shade indicates higher similarity. Through this matrix it is possible to quantitatively identify clusters of documents that share semantic elements such as context, entities, and relationships within a collection.

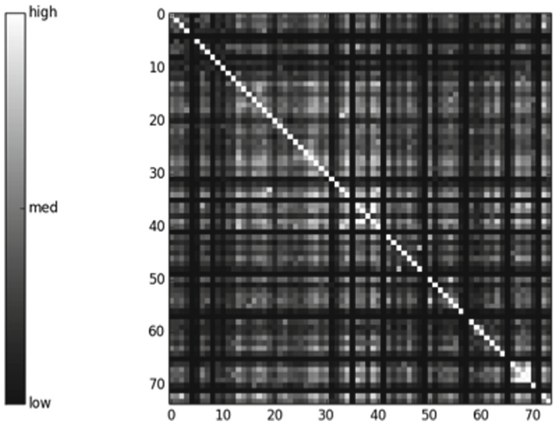


Fig. 6. Scaled similarity matrix.

For the cluster of documents with higher similarity, we performed a manual check to determine the actual similitude of semantic relationships of the data models. The manual check corroborated the high similarity of documents as well as similar entities and context in their models.

We select as an example, a pair of data models from our dataset. The models represent the publications *Plasma Vitamin E and Blood Selenium Concentrations in Norwegian Dairy Cows: Regional Differences and Relations to Feeding and Health* [42] (left) and *Lameness and Claw Lesions of the Norwegian Red Dairy Cattle Housed in Free Stalls in Relation to Environment, Parity and Stage of Lactation* [43] (right).

Figure 7 shows several relationships found in each model and between this pair of models. The models allow the detection of common entities, these are: *milk*, *cow*, and *silage*. The latter concept was found in a relationship from text of the left model and in the set of entities of the right model. Using dashed lines, we show how the common entities in the models relate to each other. The similar concepts in both models enrich and increase the semantic similarity between them. Each model has particular relationships, which are well-defined and structured. The left model has semantic annotations using relationships IsA and URI, as well as non-formal annotations using URLs and a verb. For instance, the entities *mastitis* and *paresis* contain semantic annotations to define and describe them. This model also presents a document with *URL*: http://oregonstate.edu/dept/EOARC/sites/default/files/638.pdf to explain the entity *cowrepro*. We also observe in the model on the right, that it contains semantic annotations of entities, such as *wood*, *silage*, and *pasture*. In addition, the concept *cow* uses the verb *were trimmed* to identify a relationship to text. The models show annotations of entities from a publication, DBpedia [44], and the Internet.

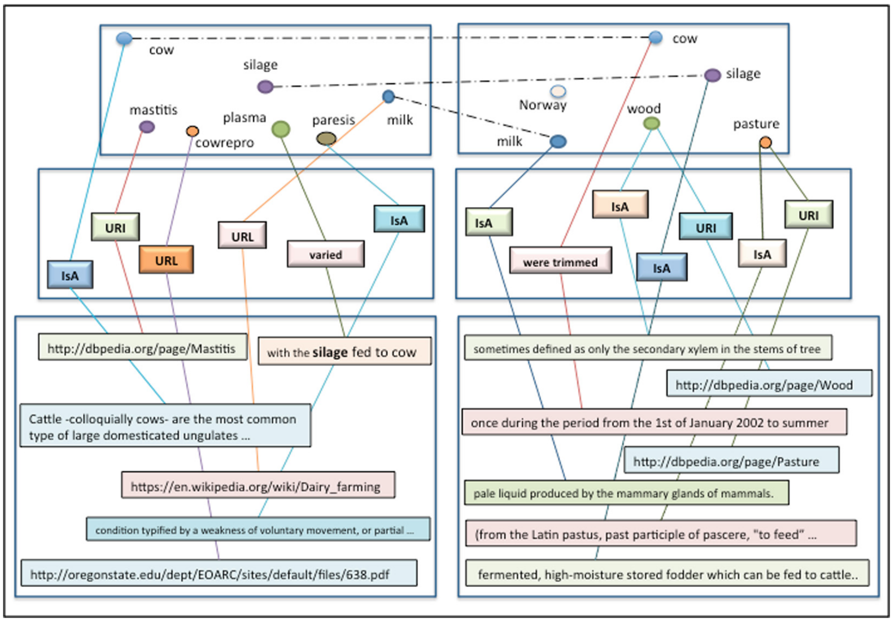

cow
silage
mastitis
plasma
paresis
milk
cowrepro
URI
URL
IsA
URL
varied
IsA
http://dbpedia.org/page/Mastitis
with the silage fed to cow
Cattle -colloquially cows- are the most common type of large domesticated ungulates ...
https://en.wikipedia.org/wiki/Dairy_farming
condition typified by a weakness of voluntary movement, or partial ...
http://oregonstate.edu/dept/EOARC/sites/default/files/638.pdf
cow
silage
Norway
wood
pasture
milk
IsA
IsA
were trimmed
URI
IsA
IsA
URI
sometimes defined as only the secondary xylem in the stems of tree
http://dbpedia.org/page/Wood
once during the period from the 1st of January 2002 to summer
http://dbpedia.org/page/Pasture
pale liquid produced by the mammary glands of mammals.
(from the Latin pastus, past participle of pascere, "to feed" ...
fermented, high-moisture stored fodder which can be fed to cattle..

Fig. 7. Relationships between two data models.

The organization of our data model facilitates finding relationships among entities and later among publications. Because the relationships can extend to more than a pair of data models, the creation of a network of related data models should be easy to accomplish with this organization.

6 Conclusion and Future Work

In this paper we present a conceptual design to generate semantic data models from digital publications systematically. The components of our data models can be used to understand and organize relevant information, synthesizing a publication. The main components of our models contain information derived from the discovery of metadata and context, categorical entities, and semantic relationships.

To organize and annotate the semantic models, we use general-purpose ontologies and a vocabulary to structure well-defined entities and semantic relationships. In particular, we match defined relationships representing association, classification, aggregation, and generalization. Therefore, capturing unambiguous semantic associations between entities. The discovery of well-defined semantic relationships in our models enable researchers to automatically exploring large collections of documents, and retrieving structured and concrete results, experimental settings, and possible hierarchical associations among concepts in a scientific environment.

The representation of our models in a machine-readable format facilitates interoperability. Keeping the provenance of a publication related to a semantic data model can help track back the source of information used to build a particular model. The provenance information is useful to ensure that researchers can access the primary source of information and investigate further an important or relevant publication.

We evaluated our approach to build semantic models through a set of experiments at different points of the process pipeline. Our approach was able to analyze documents and extract quantitative and qualitative information to compose semantic models. Our experiments show the effectiveness of our framework to recognize, enrich, and disambiguate entities, as well as to discover semantic relationships automatically. Furthermore, we compared the similarity of relationships among data models and present a visual depiction of the depth of information that can be used to compare publications automatically.

For future work, we plan to use the data models to create a semantic network. A set of models can compose a network using similar context and entities. Furthermore, we envision that the networked structure and definition of relationships can be used to compare and contrast findings and arguments within and among digital publications.

References

1. Bornmann, L., Mutz, R.: Growth rates of modern science: a bibliometric analysis based on the number of publications and cited references. J. Assoc. Inf. Sci. Technol. **66**(11), 2215–2222 (2015)
2. Peckham, J., Maryanski, F.: Semantic data models. ACM Comput. Surv. (CSUR) **20**(3), 153–189 (1988)
3. Prli, A., Martinez, M.A., Dimitropoulos, D., Beran, B., Yukich, B.T., Rose, P.W., Bourne, P.E., Fink, J.L.: Integration of open access literature into the RCSB Protein Data Bank using BioLit. BMC Bioinformatics **11**, 1–5 (2010)

4. Comeau, D.C., Islamaj Doan, R., Ciccarese, P., Cohen, K.B., Krallinger, M., Leitner, F., Lu, Z., Peng, Y., Rinaldi, F., Torii, M., Valencia, A.: BioC: a minimalist approach to interoperability for biomedical text processing. In: Database, bat064 (2013)
5. Ware, M., Mabe, M.: The STM report: an overview of scientific and scholarly journal publishing (2015)
6. The Semantic Web Science Association. http://swsa.semanticweb.org/
7. Peroni, S.: Semantic Web Technologies and Legal Scholarly Publishing. LGTS, vol. 15. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04777-5
8. Ouksel, A.M., Sheth, A.: Semantic interoperability in global information systems. ACM Sigmod Rec. **28**(1), 5–12 (1999)
9. Perez-Arriaga, M.O., Estrada, T., Abad-Mota, S.: Table interpretation and extraction of semantic relationships to synthesize digital documents. In: Proceedings of the 6th International Conference on Data Science, Technology and Applications, pp. 223–232 (2017)
10. Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr., E.R., Mitchell, T.M.: Toward an architecture for never-ending language learning. AAAI **5**, 1306–1313 (2010)
11. Nakashole, N., Weikum, G., Suchanek, F.: PATTY: a taxonomy of relational patterns with semantic types. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 1135–1145. Association for Computational Linguistics (2012)
12. Yates, A., Cafarella, M., Banko, M., Etzioni, O., Broadhead, M., Soderland, S.: TextRunner: open information extraction on the web. In Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, pp. 25–26. Association for Computational Linguistics (2007)
13. Etzioni, O., Banko, M., Soderland, S., Weld, D.S.: Open information extraction from the web. Commun. ACM **51**(12), 68–74 (2008)
14. Etzioni, O., Fader, A., Christensen, J., Soderland, S., Mausam, M.: Open information extraction: the second generation. IJCAI **11**, 3–10 (2011)
15. Fader, A., Soderland, S., Etzioni, O.: Identifying relations for open information extraction. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1535–1545. Association for Computational Linguistics (2011)
16. Hull, R., King, R.: Semantic database modeling: survey, applications, and research issues. ACM Comput. Surv. (CSUR) **19**(3), 201–260 (1987)
17. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia - a crystallization point for the Web of Data. Web Semant. Sci. Serv. Agents World Wide Web **7**(3), 154–165 (2009)
18. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. J. Am. Soc. Inf. Sci. **41**(6), 391–407 (1990)
19. Dumontier, M., Baker, C.J., Baran, J., Callahan, A., Chepelev, L., Cruz-Toledo, J., Del Rio, N.R., Duck, G., Furlong, L.I., Keath, N., Klassen, D.: The Semanticscience Integrated Ontology (SIO) for biomedical research and knowledge discovery. J. Biomed. Semant. **5**(1), 1–11 (2014)
20. Data Model - schema.org. http://schema.org/docs/datamodel.html
21. Nenkova, A., McKeown, K.: Automatic summarization. Found. Trends® Inf. Retrieval **5**(2–3), 103–233 (2011)
22. Teufel, S., Moens, M.: Summarizing scientific articles: experiments with relevance and rhetorical status. Comput. Linguist. **28**(4), 409–445 (2002)

23. Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E.D., Gutierrez, J.B., Kochut, K.: Text Summarization Techniques: A Brief Survey. arXiv preprint arXiv:1707.02268, pp. 1–9 (2017)
24. Baralis, E., Cagliero, L., Jabeen, S., Fiori, A.: Multi-document summarization exploiting frequent itemsets. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing, pp. 782–786, ACM (2012)
25. National Information Standards Organization Press: Understanding metadata. National Information Standards, vol. 20 (2004)
26. Perez-Arriaga, M.O., Wilson, S., Williams, K.P., Schoeniger, J., Waymire, R.L., Powell, A.J.: Omics Metadata Management Software (OMMS). Bioinformation **11**(4), 165172 (2015). https://doi.org/10.6026/97320630011165
27. Shinyama, Y.: PDFMiner: python PDF parser and analyzer (2015). Accessed 11 June 2015
28. Statistics - En.wikipedia.org. https://en.wikipedia.org/wiki/Wikipedia:Statistics
29. Kim, S., Han, K., Kim, S.Y. and Liu, Y.: Scientific table type classification in digital library. In: Proceedings of the 2012 ACM Symposium on Document Engineering, pp. 133–136. ACM (2012)
30. Berglund, A., Boag, S., Chamberlin, D., Fernndez, M.F., Kay, M., Robie, J., Simon, J.: XML path language (xpath). World Wide Web Consortium (W3C) (2003)
31. Perez-Arriaga, M.O., Estrada, T., Abad-Mota, S.: TAO: system for table detection and extraction from PDF documents. In: The 29th Florida Artificial Intelligence Research Society Conference, FLAIRS 2016, pp. 591–596. AAAI (2016)
32. Loria, S., Keen, P., Honnibal, M., Yankovsky, R., Karesh, D., Dempsey, E.: TextBlob: simplified text processing. Secondary TextBlob: Simplified Text Processing (2014)
33. Microsoft Cognitive Services. https://azure.microsoft.com/en-us/services/cognitive-services/bing-web-search-api
34. Zukas, A., Price, R.J.: Document categorization using latent semantic indexing. In: Proceedings 2003 Symposium on Document Image Understanding Technology, UMD, pp. 1–10 (2003)
35. Dahchour, M., Pirotte, A., Zimányi, E.: Generic relationships in information modeling. In: Spaccapietra, S. (ed.) Journal on Data Semantics IV. LNCS, vol. 3730, pp. 1–34. Springer, Heidelberg (2005). https://doi.org/10.1007/11603412_1
36. Bird, S.: NLTK: the natural language toolkit. In: Proceedings of the COLING/ACL on Interactive Presentation Sessions, pp. 69–72. Association for Computational Linguistics (2006)
37. World Wide Web Consortium. JSON-LD 1.0: a JSON-based serialization for linked data (2014)
38. JSON-LD Playground. http://json-ld.org/playground
39. Hook, V., Bark, S., Gupta, N., Lortie, M., Lu, W.D., Bandeira, N., Funkelstein, L., Wegrzyn, J., OConnor, D.T.: Neuropeptidomic components generated by proteomic functions in secretory vesicles for cellcell communication. AAPS J. **12**(4), 635–645 (2010)
40. Elmasri, R., Navathe, S.B.: Fundamentals of Database Systems. Pearson, Boston (2015)
41. Perez-Arriaga, M.O.: Automated Development of Semantic Data Models Using Scientific Publications. University of New Mexico, USA (2018)
42. Sivertsen, T., Vernes, G., Steras, O., Nymoen, U., Lunder, T.: Plasma vitamin e and blood selenium concentrations in norwegian dairy cows: regional differences and relations to feeding and health. Acta Veterinaria Scandinavica **46**(4), 177 (2005)

43. Sogstad, A.M., Fjeldaas, T., Steras, O.: Lameness and claw lesions of the norwegian red dairy cattle housed in free stalls in relation to environment, parity and stage of lactation. Acta Veterinaria Scandinavica **46**(4), 203 (2005)
44. DBpedia. http://dbpedia.org

Mining and Linguistically Interpreting Summaries from Surveyed Data Related to Financial Literacy and Behaviour

Miroslav Hudec[1](✉) and Zuzana Brokešová[2]

[1] Faculty of Economic Informatics, University of Economics in Bratislava, Dolnozemská cesta 1, Bratislava, Slovakia
miroslav.hudec@euba.sk

[2] Faculty of National Economy, University of Economics in Bratislava, Dolnozemská cesta 1, Bratislava, Slovakia
zuzana.brokesova@euba.sk

Abstract. Financial decisions represent important decisions in everyday life, as they could affect the financial well-being of the individuals. These decisions are affected by many factors including level of financial literacy, emotions, heuristics and biases. This paper is devoted to mining and interpreting information regarding effect of financial literacy on individuals' behaviour (angst, fear, nervousness, loss of control, anchoring and risk taking) from the data surveyed by questionnaire applying linguistic summaries. Fuzzy sets and fuzzy logic allow us to mathematically formalize linguistic terms such as *most of*, *high literacy*, *low angst* and the like, and interpret mined knowledge by short quantified sentences of natural language. This way is suitable for managing semantic uncertainty in data and in concepts. The results have shown that for the majority of respondents having low level of financial literacy, angst and other treats represent serious issues, as expected. On the other hand, about half of respondents with high level of literacy do not consider these treats as significant. This effect is emphasized by the experimenting with socio-demographic characteristics of respondents. This research has also observed problems in applying linguistic summaries on data from questionnaires and suggests some recommendations.

Keywords: Financial literacy · Emotions · Biases
Quantified sentence of natural language
Flexible data summarization · Fuzzy logic · Questionnaire

1 Introduction

According to the standard economic theory, individuals make purely rational economic decisions with the main aim of utility maximization. They use all available information, and they are able to process them in a very effective way. However, real world observations do not always support these normative approach assumptions. Descriptive models conclude that humans' psychology plays

J. Filipe et al. (Eds.): DATA 2017, CCIS 814, pp. 67–83, 2018.
https://doi.org/10.1007/978-3-319-94809-6_4

an important role in the decision-making process, and individuals' economic decisions are often driven by their emotions, heuristics and biases (e.g. [2,6,14,32]). In addition, for sound financial decisions, adequate level of the financial literacy was identified as an important factor [21].

To our best knowledge, the effect of financial literacy to cognitive heuristics and biases has not been widely examined in the literature. Hence, in the paper, we have examined the effect of financial literacy level to (i) feeling like angst, nervousness, loss of control and fear regarding possible catastrophic scenarios; (ii) how these people are influenced by anchor questions; (iii) how they decide about risky investments.

The focus of our work is on mining this effect from the data surveyed by questionnaires in an easily understandable and interpretable way, and interpreting obtained results from the behavioural perspective. First, summaries from the data are better understandable if they are not as terse as numbers [41]. Secondly, there are often uncertainties in answers, which should not be neglected [11,34]. Thirdly, answers to respective questions in the realized survey are numbers and categorical data.

Fuzzy sets and fuzzy logic allow us to mathematically formalize linguistic terms such as *most of*, *low literacy*, *high angst* and the like [42]. They make possible partial membership degrees of elements near the borderline cases to these concepts. Works of, e.g. Keefe [16] and Wright [37] suggest that it is not possible to operate with a language free of vagueness (acceptance of borderline cases, lack sharp boundaries between concepts and sorties paradoxes). This observation also holds for surveys and subsequent mining information from the collected data.

Concept known as Linguistic Summaries (LSs) is a promising way for these tasks. Generally, LSs summarize the whole data set, or a restricted part into concise short sentences of natural language. In the latter, LSs are of the structure Q R *entities are (have)* S, where Q is a relative quantifier, S is a summarizer and R puts some restriction on a data set. One example of such a summary is: *most of young respondents have high financial literacy.*

In order to reveal influence of financial literacy to behaviour, we have conducted survey among Slovak population and adjusted LSs to mine and interpret relational knowledge related to financial literacy and respective attributes from the collected data. Preliminary results of this research were published in position paper at the DATA 2017 conference [12]. This paper provides deeper insight into this topic, extends mining summaries from the data with compound predicates in the restriction part in order to reveal further dependencies among attributes, and examine benefits and limitations in mining linguistic summaries from the data gathered by questionnaires.

This chapter is organized as follows. Section 2 gives some preliminaries of linguistic summaries, which are used throughout this paper. Section 3 is devoted to mining linguistic summaries from the collected data and discussion related to obtained results. Consecutive Sect. 4 elaborates issues of applying linguistic summaries on questionnaires' data and speculates possible further research tasks and opportunities. Finally, Sect. 5 presents a summary of the paper.

2 Preliminaries of Linguistic Summaries and Their Quality

This section provides structures and equations which are used in mining and interpreting summaries throughout this paper.

2.1 Linguistic Summaries and Their Structures

LSs have been initially introduced by Yager [40] in order to express summarized information from the data by linguistic terms instead of numbers. Since then, LSs have been theoretically improved and applied in a variety of fields. An overview of recent developments can be found in [1].

In this paper the focus is on LSs of the structure *Q R entities in a data set are (have) S*, developed by Rasmussen and Yager [27,28], which are suitable for mining and interpreting relational knowledge among attributes. One example of a possible summary is *most of low financially literate people feel high level of angst*. The validity (truth value) is computed as

$$v(Qx(P(x))) = \mu_Q(\frac{\sum_{i=1}^{n} t(\mu_S(x_i), \mu_R(x_i))}{\sum_{i=1}^{n} \mu_R(x_i)}) \tag{1}$$

where n is the cardinality of elements in a data set (in our research the number of respondents), $\frac{\sum_{i=1}^{n} t(\mu_S(x_i), \mu_R(x_i))}{\sum_{i=1}^{n} \mu_R(x_i)}$ is the proportion of the records in a data set that meet the S and belong to the R, t is a t-norm, μ_S, μ_R, and μ_Q are membership functions explaining summarizer S, restriction R and relative quantifier Q, respectively.

The validity can be also expressed by scalar cardinality as [24]

$$v(Qx(P(x))) = \mu_Q(\frac{\text{card}(S \cap R)}{\text{card}(R)}) \tag{2}$$

where $\sum_{i=1}^{n} t(\mu_S(x_i), \mu_R(x_i))$ is represented by the scalar cardinality of the intersection between S and R.

The validity v gets value from the unit interval. If v is closer to the value of 1, then the relation between R and S explained by Q is more significant. The goal of LSs is to reveal such relations. LSs are graphically illustrated in Fig. 1, where grey areas between sets *low*, *medium* and *high* emphasizes the uncertain area, i.e. area where unambiguous belonging to a particular set cannot be arranged.

Flexible summarizers, restrictions and quantifiers are mathematically formalized by fuzzy sets [27]. For instance, fuzzy set *around anchor value of m* can be expressed by symmetric triangular fuzzy set (Fig. 2) where belonging to the set decreases when distance to the anchor m increases, or mathematically

$$\mu_A(x) = \begin{cases} 1 & \text{for } x = m \\ \dfrac{x-a}{m-a} & \text{for } x \in (a, m) \\ \dfrac{b-x}{b-m} & \text{for } x \in (m, b) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

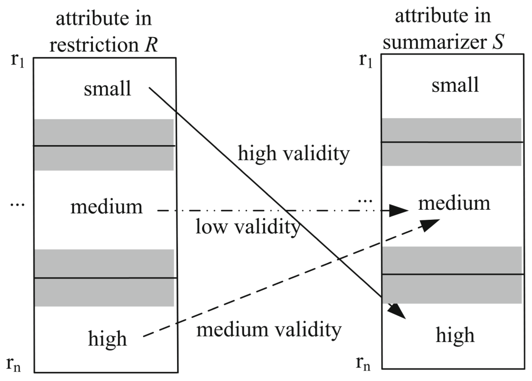


Fig. 1. Graphical illustration of LSs with restriction.

The benefit against interval is in the intensity of belonging to a set. The closer is element to boundary, the lower matching degree it has. Thus, defining boundaries is not as sensitive as for classical intervals. Accordingly, fuzzy sets expressing concepts *small* and *high* are plotted in Fig. 3.

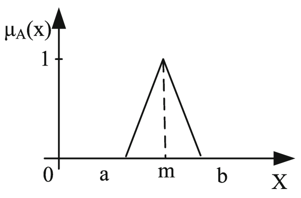


Fig. 2. Triangular fuzzy set *around m*.

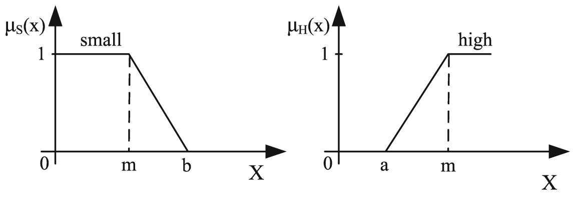


Fig. 3. Fuzzy set expressing concepts *small* and *high*.

For categorical data (or domains containing small number of elements) matching degree to a fuzzy set FL is directly assigned to each element, i.e. [18]

$$FL = \{(x_1, \mu(x_1)), ..., (x_n, \mu(x_n))\} \tag{4}$$

Restriction R, as well as summarizer S, could be atomic or composed of several atomic predicates. The role of restriction is to focus on the particular flexible subset of a data set. When R consists of two or more atomic predicates, they should be merged by *and* connective to focus on a more specific subset of a considered data set. An example of such summary is *most of high financially literate and young people have low level of angst*. This case is illustrated in Fig. 4. Regarding summarizer S, atomic predicates might be merged by *and* or by *or* connective depending on the particular task.

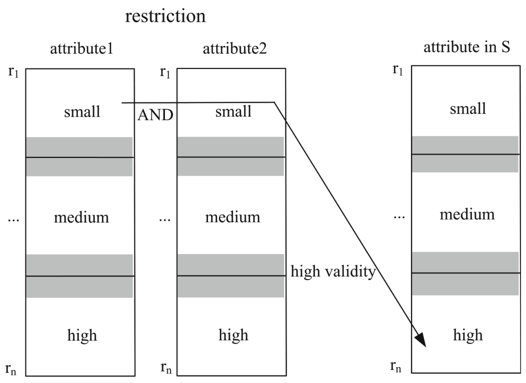


Fig. 4. Summary consisted of two attributes in restriction part.

When restriction or summarizer consists of several atomic predicates connected by the *and* connective or conjunction, t-norms come to the stage. All t-norms meet axiomatic properties, but differ in satisfying algebraic ones such as idempotency, nilpotency and limit property [17]. According to [10] only minimum t-norm is suitable as a connective among atomic predicates in R and S, for ensuring correct proportions as well as in (1), for ensuring considering only relevant subset of the data. Similarly, for *or* connective only maximum t-conorm is suitable, because it is dual to the minimum t-norm and therefore shares the same properties.

2.2 Quality of Linguistic Summaries

The validity v of LSs with restriction (1) is not the suitable measure, because it might be influenced by outliers or low data coverage [10,39]. Several quality measures were suggested in, e.g. [8,25]. Due to their complexity and partial overlapping, simplified quality measure merging validity and data coverage is suggested in [10]

$$Q_c = \begin{cases} t(v, C) & \text{for } C \geq 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where C is data coverage and t is a t-norm. In this equation we need a non-idempotent t-norm, e.g. product one. Data coverage is calculated form the coverage index

$$i_c = \frac{\sum_{i=1}^{n} t(\mu_S(x_i), \mu_R(x_i))}{n} \tag{6}$$

where parameters have the same mining as in (1) by transformation [39]

$$C = f(i_c) = \begin{cases} 0, & \text{for } i_c \le r_1 \\ 2(\frac{i_c - r_1}{r_2 - r_1})^2, & \text{for } r_1 < i_c \le \frac{r_1 + r_2}{2} \\ 1 - 2(\frac{r_2 - i_c}{r_2 - r_1})^2, & \text{for } \frac{r_1 + r_2}{2} < i_c < r2 \\ 1, & \text{for } i_c \ge r_2 \end{cases} \tag{7}$$

where $r_1 = 0.02$ and $r_2 = 0.15$, because in LSs with restriction only small subset of data is included in both R and S (Fig. 1). Anyway, parameters r_1 and r_2 can be adjusted according to the number of atomic predicates in restriction.

3 Mining Relational Knowledge Among Attributes of Financial Behaviour

This section explains procedures and results presented in [12] as well as new ones, which are a natural continuation of already achieved ones.

The first step consists of constructing membership functions for attributes' linguistic labels appearing in summaries.

Attribute *financial literacy level* is used in all experiments. The level of financial literacy was not included in the questionnaire, because people might over or underestimate it. Thus, value of this level was aggregated from the answers to several questions. Financial literacy is expressed on the [0, 5] scale, where 0 is the lowest and 5 the highest level. Other questions were directly focused on feelings to potential kinds of threats, willingness to participate in financial games and influences by anchor questions as well as on educational level, salary and gender.

In order to apply linguistic summaries the *financial literacy level* attribute is fuzzified into three fuzzy sets: *low literacy* (LL), *medium literacy* (ML) and *high literacy* (HL) using notation (4) in the following way

$HL = (5, 1), (4, 0.75), (3, 0.45)$
$ML = (1, 0.35), (2, 1), (3, 1), (4, 0.35)$
$LL = (0, 1), (2, 0.75), (3, 0.45)$

It means that level 5 is without any doubt high, level 4 significantly high but not fully and level 3 is partially high. Other two levels do not belong to HL set. The recognized drawback in fuzzification is for ML fuzzy set. It contains two elements with matching degree equal to 1. Hence, higher numbers of elements belong to this set. The better option is using scale consisting odd number of options.

3.1 Relation Between Financial Literacy and Attributes of Fear, Angst, Nervousness and Loss of Control

In this section, we recall already mined summaries in [12] related to attributes *fear*, *angst*, *nervousness* and *loss of control* in order to emphasize results and elaborate further experiments.

Experiments

Emotional heuristics and biases including fear, angst, nervousness or loss of control could significantly shape financial decision [20]. For example, in the case of insurance purchase, Zaleskiewicz et al. [43] identified increase in the flood insurance demand after the recent flood. The role of financial literacy in this relation is questionable and therefore should be further examined.

Respondents in our survey were asked to express their emotions (angst, fear, nervousness and loss of control) evoked by imagination of possible negative events that could affect their life. This imagination was visualized in the pictures of unexpected natural disaster, death of relatives, war and deadly epidemic. Based on this imagination, respondents were asked about their level of fear, angst, nervousness or loss of control. Possible answers were: not at all, weakly, moderately, intensely and very intensely. These categorical attributes are fuzzified into sets *low* and *high* by (4):

$$L = (\text{not at all}, 1), (\text{weakly}, 0.75), (\text{moderately}, 0.45)$$
$$H = (\text{very intensely}, 1), (\text{intensely}, 0.75), (\text{moderately}, 0.45)$$

In order to reveal summaries, we have calculated validity of each summary by (1) and examined their respective qualities by (5).

In the first experiment, we were focused on evaluation summaries: *most of respondents of low financial literacy have high values of respective treats* in order to reveal, whether low financial literacy decreases quality of life. Mined results are shown in Table 1. Financial literacy is a restriction (R) and summarizers (S) are considered treats, respectively. First row in Table 1 means that LS *most of respondents with low financial literacy have high angst* has validity almost 1, whereas the sentence explained by quantifier *about half* has validity almost 0.

Table 1. Result of summaries Q respondents of low financial literacy have high values of respective treats [12].

Low financial literacy		
High	Validity for quantifier (1)	Coverage (7)
Angst	Most of −0.9996, about half 0.0004	1
Loss of control	Most of −0.5863, about half 0.4137	1
Fear	Most of −0.5818, about half 0.4182	1
Nervousness	Most of −0.7674, about half 0.2326	1

Mined sentences have shown that the majority of respondents with low financial literacy feel high level of fear, angst, loss of control and nervousness. The most problematic attribute is angst. These people might think that potential dangerous situation, if appear, will significantly devalue their lives and properties.

On the other side, majority of people with high financial literacy does not have straightforwardly low intensities of these treats as is illustrated in Table 2. In fact, about half of them do not consider examined treats significant. The explanation of our results, regarding the individuals with high financial literacy and their levels of emotions, could be the effect of socio-demographic factor. The questions about the role of gender, education or income are open. Researches indicate that higher level of financial literacy is strongly and positively correlated with higher level of education and income (e.g. [22,23]). In addition, differences in the level of financial literacy among genders are supported by the previous research as well. Moreover, BucherKoenen et al. [3] concluded that women are less likely than men to answer correctly in financial literacy questions and they are more likely to indicate that they do not know the answer. In self assessment, they also indicate lower values of their financial literacy compare to men. These factors could therefore have an effect on the level of emotional threats.

Table 2. Result of summaries Q respondents of high financial literacy have low values of respective treats [12].

High financial literacy		
Low	Validity for quantifier (1)	Coverage (7)
Angst	About half −0.7129, few −0.1411	0.7888
Loss of control	About half 1	1.0000
Fear	About half 1	0.9988
Nervousness	About half 1	0.9346

We decided to test the effect of three socio-demographic factors (income, education and gender), in order to shed more light to the results of our experiment shown in Table 2.

Experimenting with the Role of Other Attributes

Additional attributes, which might reveal influence reaching low intensities of considered threats from our questionnaire are income, education and gender.

The first attribute was income. We wished to reveal, whether majority of respondents of high financial literacy and high income do not consider studied treats so problematic, i.e. high financial literacy and income (knowledge and better financial situation) cause decreased relevance of these threats.

Answer to the question regarding income was one of options: [up to 300], [301, 600], [601, 800], [801, 1 000], [1 001, 1 500], [more than 1 500]. Fuzzy set *high income* is expressed as follows

$$HI = \{([\text{more than } 1\,500)], 1), ([1\,001, 1\,500], 0.75), ([801, 1\,000], 0.25)\}$$

i.e. income in interval [801, 1000] partially belongs to the set HI.

Adding attribute in the restriction part in (1) causes that the smaller data subset is enveloped and therefore we relaxed parameters r_1 and r_2 in (7) to values 0.01 and 0.08, respectively. Results of this experiment are shown in Table 3. Slight improvements were recognized for angst and lost of control threats.

The results regarding the role of income in the relation between financial literacy support our assumption. In case of angst and loss of control, the perception by the individuals decrease with the increase in income. Intuitively, those with higher income and therefore higher standard of living should also have a higher level of financial literacy. This is supported by the literature as well [22]. They have to do many financial decisions on a daily basis. With increasing income, also the complexity of these financial decisions increase and higher level of financial literacy is inevitable. Higher income is also related to higher living standard. On the one hand, higher income individuals could lost more that those without tangible and financial assets. But, they have much more possibilities to secure themselves and their families.

Table 3. Result of summaries Q respondents of high financial literacy and high income have low values of respective treats.

High financial literacy and high income		
Low	Validity for quantifier (1)	Coverage (7)
Angst	About half −0.8901, few −0.1099	0.9522
Loss of control	About half 0.9495, most of −0.0505	1.0000
Fear	About half 1	1.0000
Nervousness	About half 1	0.9980

The second considered attribute was educational level. One might expect that people with high education and high financial literacy consider such treats as minor. Fuzzy set *high education* is the following

$$HE = (\text{doctoral level}, 1), (\text{master level}, 0.75), (\text{bachelor level}, 0.25)$$

The other two options: secondary level and elementary level do not belong to this fuzzy set.

Result are in Table 4. In this experiment, significant decrease of quality of life regarding angst is recorded, whereas for other treats changes were not recognized.

The positive correlation among the higher level of financial literacy and higher level of income is obvious and also supported by the previous research [22]. However, effect of education and financial literacy on the level of emotional state of individual is not widely researched. Our analysis brings an interesting result regarding their effect on angst. Based on our results, we could conclude that respondents with higher level of education and higher financial literacy suffer more regarding the angst. This results could be explained in two ways. Firstly, the higher education do not have to be necessarily correlated with high income (for example teachers in elementary, high school and universities are well educated but not well paid in Slovakia) and therefore their level of angst and frustration could be higher. The second explanation could be the fact that higher educated individuals are more reflective to their abilities, as well as, they are usually more open minded which could increase they angst.

Table 4. Result of summaries Q respondents of high financial literacy and high education have low values of respective treats.

High financial literacy and high education		
Low	Validity for quantifier (1)	Coverage (7)
Angst	About half −0.5864, few −0.4136	0.5008
Loss of control	About half 1	0.9986
Fear	About half 1	0.9343
Nervousness	About half 1	0.7953

Next experiment was focused on the gender influence to threats for highly literate people. In this experiment restriction for gender is a sharp condition. Mined validities for respective sentences are in Table 5.

The difference among the economic decisions between male and female is widely discussed in economic literature, e.g. [4,19,26,30]. Females are usually understand to be more risk averse and less confident in economics decisions. In general, females are considered to be more emotional than males. Our results supported these assumptions. The differences are significant in case of angst and nervousness, where few highly financially literate females testify much lower level of angst and nervousness compared to males. It is important to emphasize that we are speaking about the females with high level of financial literacy and the effect on gender in this emotions is evident. The differences regarding loss of control and fear are not significant but in majority of them females show lower level of emotions.

Table 5. Result of summaries Q respondents of high financial literacy and gender have low values of respective treats.

High financial literacy and gender				
	Male		Female	
Low	Validity for quantifier	Coverage	Validity for quantifier	Coverage
Angst	About half −1	0.9555	Few −0.8971, about half −0.1029	0.8288
Loss of control	About half 0.8052, few −0.1948	1.0000	About half 1	0.9971
Fear	About half 1	1.0000	About half 0.8505, few −0.1945	0.8134
Nervousness	About half 1	0.9785	About half 0.6396, few −0.3604	0.6786

Besides the relation between financial literacy and emotions, financial decision making of individuals could be affected by other biases and heuristic defined in behaviour economic. We focus on two specific variables: anchoring and risk taking. The results of our experiments are in the next two successive sections (Sects. 3.2 and 3.3).

3.2 Relation Between Financial Literacy and Answer to Anchor Question

Anchoring represents a subconscious tendency of individuals to rely on onward information in the decision-making [6]. This information does not have to be necessarily important for the decision. However, research [5] provides an evidence that it could strongly bias individual's decision. The role of financial literacy in the process of anchoring has not been previously studied. The open question is, whether are more financially literate persons more sensitive to anchoring?

In this experiment, we worked with numerical attribute: number of inhabitants in Iowa[1]. Anchor value was set to 1 500 000 inhabitants. In order to reveal behaviour of respondents who have answered around this anchor value, we relaxed crisp value 1 500 000 to the fuzzy set *around 1 500 000*, which is a convex triangular fuzzy set with limited support. This set is represented by (3) with the following values of parameters: $a = 1200000$, $m = 1500000$ and $b = 1800000$.

In this experiment, significant relations between literacy and answers to anchor questions have not been recorded by LSs. From the opposite view (restriction is anchor question and summarizer is financial literacy), we recorded high validity for the summary *most of respondents with answer around anchor have medium literacy*. These relations are summarized in Table 6.

We could conclude that individuals with high level of financial literacy do not reflect the anchor in their decision-making. However, also respondents with low level of financial literacy do not react to anchoring. Anyway, this experiment requires further analysis. Different terms (fuzzy sets) regarding answers, for instance, *far from 1 500 000*, *significantly lower than 1 500 000*, *significantly higher than 1 500 000* and the like might be promising for further experiments.

Table 6. Result of summaries Q respondents with answer around anchor value have (low, medium, high) values of financial literacy [12].

Number of answers around the anchor value		
Low	Validity for quantifier (1)	Coverage (7)
Low financial literacy	Few 0.5686, about half 0.4314	0.1249
Medium financial literacy	Most of 1	0.9456
High financial literacy	Few -0.3018, about half 0.6981	0.1918

[1] In the literature, different types of anchor are used. For more information see [5].

An approach based on fuzzy functional dependencies, mentioned in Sect. 4, might be beneficial in this direction.

3.3 Relation Between Financial Literacy and Risk Taking

Risk aversion is an important parameter in financial decisions as majority of these decisions include risk. In the population, majority of people was identified as risk averse, which means that these individuals prefer safe option against the risky one [15]. Previous research identified that the level of risk aversion could be influenced by socio-demographic characteristics (for example age or gender) as well as experiences [7]. For example, investment bankers reveal lower level of aversion to risk in their investment decisions due to their experiences [38]. Experience could therefore represent a important driver of the level of risk taking. As higher level of financial literacy could be understand as a proxy of more experiences in financial decision making. We decided to test this assumption by linguistic summaries. In economic literature, risk taking is often measured by the decisions in the lottery, where individuals could choose between sure and risky alternative, e.g. [9].

In our experiments, participation in lottery is numerical attribute. Respondents were asked how much they are willing to pay for the participation in the lottery, where one out of 10 players win 1000 EUR. Based on the expected value theory, those who are willing pay less than 100 EUR could be considered as risk averse, those who would like to pay 100 EUR are assumed as risk neutral and those who will pay more than 100 EUR are recognized as risk loving. The lowest recorded value was 0 and the highest 333, whereas mean value was 21.30 EUR. Data distribution is not uniform, which means that we cannot create three granules: *low*, *medium* and *high* by uniformly covering domain. The options are statistical mean based method [33] or logarithmic transformation [13]. Applying the former, fuzzy set *low participation* is expressed by parameters

Table 7. Result of summaries between financial literacy and participation in lottery using quantifiers *few*, *about half* and *most of* [12].

High Financial Literacy		
	Validity for Q	Coverage
Low participation	Most of 1	1.0000
High participation	Few −1	$0.0081 < 0.5$
Low financial literacy		
Low participation	Most of 1	1.0000
High participation	Few −1	0.0000
Medium financial literacy		
Low participation	Most of 1	1.0000
High participation	Few −1	$0.2898 < 0.5$

$m = 25$ and $b = 37.5$, whereas set *high participation* is expressed by parameters $a = 62.5$ and $m = 75$ (Fig. 3). Results from this experiment are summarized in Table 7.

We have recorded that respondents belonging to all three categories of financial literacy prefer low participation in lottery, which supported the prevailing risk aversion character of the population. We have not identified effect of financial literacy on risk taking. Based on our results, we could not support the premise about the effect of financial literacy on the risk taking level.

4 Discussion and Further Perspectives

Our paper represents, according our best knowledge, the first attempt to analyse impact of financial literacy on emotions and cognitive biases as well as to analyse these data with linguistic summaries.

Our contribution to the literature could be defined in two areas: related to applying linguistic summaries and related to economics. From the economic point of view, our findings have shown that feelings and emotions are related with the level of financial literacy of individuals. Those with low level of financial literacy reveal high level of angst, nervousness, fear and loss of control. The relation between sensitivity to anchoring and the level of risk taking do not differ based on the level of financial literacy. As the effect of anchor on decision is supported by the literature, one could speculate that those with high level of financial literacy notice the anchor but isolate its effect in decision-making. Those with low levels of financial literacy presumably do not notice anchor.

Regarding the LSs, answers to majority of questions were categorical ones. Some of them are expressed through intervals (e.g. age and income) instead of numbers. We have been able to fuzzify intervals of the attribute *income* into the fuzzy sets (Sect. 3.1). The problem was fuzzification of attribute *age*. Answer to question regarding the respondent's age was one of the following intervals: [18, 24], [25, 34], [35, 44], [45, 54], [55, 64]. Thus, we cannot create intuitive and continuous fuzzy sets for granules *young* (decreasing function), *middle aged* (trapezoidal function) and *old* (increasing function) as it is case when respondents are asked to provide their exact age.

Furthermore, when respondent selects one interval we do not know, whether value is in the middle of interval or near the border. Thus, two similar values might be differently treated. On the other hand, respondents cannot always provide exact value. A possible solution are fuzzy numbers, where respondent can explain answer as: *the most likely m but for sure smaller than a and not higher than b*, where a, m and b are real numbers. Such answers can be recorded in any relational database management system extended with the fuzzy meta model [11,31] for further data analyses.

Concerning categorical data, respondents are not always sure which of the predefined linguistic category (very intensely, intensely, moderately, etc.) is the best option. One possible solution is expressing respondents' answers by the hesitant fuzzy linguistic element from a hesitant fuzzy linguistic term set.

A hesitant fuzzy linguistic term set is an ordered finite subset of the consecutive linguistic terms: $S = \{s_0, ..., s_\tau\}$ [29]. In this way answers might be: at least intensely, between weakly and intensely, not higher than moderately. This might be comprehensive way to offer more flexibility in expressing feelings, but further research is advisable especially from the data mining perspective.

We were not able to evaluate summaries consisted of three atomic attributes in the restriction part aggregated by *and* connective: *Q respondents with high education and high income and high financial literacy have low angst (and other treats)* due to low data coverage caused by the data sample size. Thus, for such summaries larger number of respondents is required to reach sufficient data coverage, i.e. to avoid conclusions form a small data subset. Initially, linguistic summaries were developed for mining relational knowledge form large data sets.

The structure of questions and task for mining relational knowledge reveals that the methodology based on Fuzzy Functional Dependencies (FFD) could be also suitable. Roughly speaking, FFD reveal, whether records (in our case respondents) that have similar values in one attribute have also similar values in another attribute applying similarity functions and further calculations of conformance and implication. If a similarity holds for majority of records, then there is a strong functional dependency which might be linguistically interpreted [35]. An overview of FFD can be found in, e.g. [36]. In this way, we could cover mining relational knowledge from questionnaires by two approaches capable to manage semantic uncertainty. Hence, results from this research should be in the near future confronted with approach based on FFD.

Finally, regarding the future socio-economic research, our results reveal the link among the financial literacy and emotions as well as psychological heuristic and biases for Slovak respondents. In addition, we tested the effect of main socio-demographic factors (gender, income, education) in this relation. Promising aspect for research is comparison with other countries as well as application of various methods for financial literacy measurement. To reveal such answers an international survey and research is advisable.

5 Conclusion

Financial literacy, its improvements and its drivers is a relevant topic, which should be analysed. Sound financial decision could positively affect individuals' well-being and living standard. In this paper, we have shown that mining relational knowledge from well-designed questionnaires by fuzzy logic is a valuable contribution to this field. By fuzzy logic in general and linguistic summaries in particular, we are able to recognize significant relations between restriction and summarizer explained through relative quantifiers.

Data mining results have shown that low financial literacy indicates that people have lower quality of life, due to high level of angst, nervousness, fear and loss of control. Although, majority of respondents with high level of financial literacy do not have straightforwardly low angst and other threats, at least about half of them do not consider the treats significant. For respondents with low level of financial literacy these treats represent problems. On the other side, high level of financial literacy does not straightforwardly mean big improvement. This effect was emphasized by incorporating socio-demographic characteristics of respondents (gender, education, income) into our analysis.

Besides the emotional factors, we also have studied the relation between the level of financial literacy and anchoring. Reaction to anchoring is low in the group with high level of financial literacy and low level of financial literacy as well. Several studies supported the role of anchor in decision-making. In our study, the absence of this effect on the two extremes of financial literacy (high and low) brings an interesting finding to the literature. Furthermore, we have recorded that respondents belonging to all three categories of financial literacy show risk aversion and in lottery question prefer safe option to the risky one. It is an expected result as the majority of the population is risk averse, according to previous research. Our results suggest that risk aversion do not vary with the level of financial literacy.

Our research has recorded few limitations. Linguistic summaries were initially developed for mining relational knowledge from large data sets. In this research, we applied them on numerical and categorical data from questionnaires. Benefits are in linguistically interpreted short quantified sentences. For more complex summaries (several attributes in the restriction part), higher number or respondents and/or predominance of numeric answers are required. For the future tasks, we would like to increase the number of respondents from countries with different socio-demographic background to support our findings. Regarding the data mining approach, in the future research, we intent to apply powerful, but also computationally demanding approach for revelling flexible functional dependencies among attributes to confront already reached results.

Acknowledgements. This paper is part of a research grant VEGA No. 1/0849/15 entitled Economic and social aspects of the information asymmetry in the insurance market supported by the Ministry of Education, Science, Research and Sport of the Slovak Republic. The authors would especially like to thank Monika Jurkovičová and Erika Pastoráková who collected the data used in this paper.

References

1. Boran, F.E., Akay, D., Yager, R.R.: An overview of methods for linguistic summarization with fuzzy sets. Expert Syst. Appl. **61**, 356–377 (2016)
2. Borghans, L., Duckworth, A.L., Heckman, J.J., Ter Weel, B.: The economics and psychology of personality traits. J. Hum. Resour. **43**(4), 972–1059 (2008)
3. BucherKoenen, T., Lusardi, A., Alessie, R., Van Rooij, M.: How financially literate are women? An overview and new insights. J. Consum. Aff. **51**(2), 255–283 (2017)
4. Eckel, C.C., Grossman, P.J.: Men, women and risk aversion: experimental evidence. In: Plott, C.R., Smith, V.L. (eds.) Handbook of Experimental Economics Results, vol. 1, pp. 1061–1073. Elsevier, Amsterdam (2008)
5. Furnham, A., Boo, H.C.: A literature review of the anchoring effect. J. Soc. Econ. **40**, 35–42 (2011)

6. Gilovich, T., Griffin, D., Kahneman, D.: Heuristics and Biases: The Psychology of Intuitive Judgement. Cambridge University Press, Cambridge (2002)
7. Halek, M., Eisenhauer, J.G.: Demography of risk aversion. J. Risk Insur. **68**, 1–24 (2001)
8. Hirota, K., Pedrycz, W.: Fuzzy computing for data mining. Proc. IEEE **87**, 1575–1600 (1999)
9. Holt, C.A., Laury, S.K.: Risk aversion and incentive effects. Am. Econ. Rev. **92**, 1644–1655 (2002)
10. Hudec, M.: Merging validity and coverage for measuring quality of data summaries. In: Kulczycki, P., Kóczy, L.T., Mesiar, R., Kacprzyk, J. (eds.) CITCEP 2016. AISC, vol. 462, pp. 71–85. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-44260-0_5
11. Hudec, M.: Storing and analysing fuzzy data from surveys by relational databases and fuzzy logic approaches. In: XXVth IEEE International Conference on Information, Communication and Automation Technologies (ICAT 2015), Sarajevo, pp. 220–225 (2015)
12. Hudec, M., Brokešová, Z.: Mining and linguistically interpreting data from questionnaires. In: 6th International Conference on Data Science, Technology and Applications (DATA 2017), Madrid, pp. 249–254 (2017)
13. Tou, J.T.: Information systems. In: von Brauer, W. (ed.) GI 1973. LNCS, vol. 1, pp. 489–507. Springer, Heidelberg (1973). https://doi.org/10.1007/3-540-06473-7_52
14. Kahneman, D.: Maps of bounded rationality: psychology for behavioral economics. Am. Econ. Rev. **93**(5), 1449–1475 (2003)
15. Kahneman, D., Tversky, A.: Choices, values, and frames. Am. Psychol. **39**(4), 341–350 (1984)
16. Keefe, R.: Theories of vagueness. Cambridge University Press, Cambridge (2000)
17. Klement, E.P., Mesiar, R., Pap, E.: Semigroups and triangular norms. In: Klement, E.P., Mesiar, R. (eds.) Logical, Algebraic, Analytic, and Probabilistic Aspects of Triangular Norms, pp. 63–94. Elsevier, Amsterdam (2005)
18. Klir, G., Yuan, B.: Fuzzy Sets and Fuzzy Logic, Theory and Applications. Prentice Hall, Upper Saddle River (1995)
19. Laasch, O., Conaway, R.: Gender differences in preferences. J. Econ. Lit. **47**(2), 448–474 (2009)
20. Lee, C.J., Andrade, E.B.: Fear, excitement, and financial risk-taking. Cogn. Emot. **29**, 178–187 (2015)
21. Lusardi, A.: Financial literacy: an essential tool for informed consumer choice? National Bureau of Economic Research, Working Paper 14084, Cambridge, MA (2008)
22. Lusardi, A., Mitchell, O.S.: The economic importance of financial literacy: theory and evidence. J. Econ. Lit. **52**(1), 5–44 (2014)
23. Lusardi, A., Mitchell, O.S.: Financial literacy around the world: an overview. J. Pension Econ. Financ. **10**(4), 497–508 (2011)
24. Niewiadomski, A., Ochelska, J., Szczepaniak, P.S.: Interval-valued linguistic summaries of databases. Control Cybern. **35**, 415–443 (2006)
25. Pereira-Fariña, M., Eciolaza, L., Triviño, G.: Quality assessment of linguistic description of data. In: ESTYLF, Valladolid, pp. 608–612 (2012)
26. Powell, M., Ansic, D.: Gender differences in risk behaviour in financial decision-making: an experimental analysis. J. Econ. Psychol. **18**(6), 605–628 (1997)
27. Rasmussen, D., Yager, R.R.: Summary SQL - a fuzzy tool for data mining. Intell. Data. Anal. **1**, 49–58 (1997)

28. Rasmussen, D., Yager, R.R.: Finding fuzzy gradual and functional dependencies with SummarySQL. Fuzzy Sets Syst. **106**, 131–142 (1999)
29. Rodríguez, R.M., Martínez, L., Herrera, F.: Hesitant fuzzy linguistic terms sets for decision making. IEEE Trans. Fuzzy Syst. **20**, 109–119 (2012)
30. Sunden, A.E., Surette, B.J.: Gender differences in the allocation of assets in retirement savings plans. Am. Econ. Rev. **88**(2), 207–211 (1998)
31. Škrbić, S., Racković, M., Takači, A.: Towards the methodology for development of fuzzy relational database applications. Comput. Sci. Inf. Syst. **8**, 27–40 (2011)
32. Tversky, A., Kahneman, D.: Judgement under uncertainty: heuristics and biases. Sci. New Ser. **185**(4157), 1124–1131 (1975)
33. Tudorie, C.: Qualifying objects in classical relational database querying. In: Galindo, J. (ed.) Handbook of Research on Fuzzy Information Processing in Databases, pp. 218–245. Information Science Reference, Hershey (2008)
34. Viertl, R.: Fuzzy data and information systems. In: 15th International WSEAS Conference on Systems, Corfu, pp. 83–85 (2011)
35. Vucetic, M., Hudec, M., Vujoev, M.: A new method for computing fuzzy functional dependencies in relational database systems. Expert Syst. Appl. **40**, 2738–27450 (2013)
36. Vueti, M., Vujoevi, M.: A literature overview of functional dependencies in fuzzy relational database models. Tech. Technol. Educ. Manag. **7**, 1593–1604 (2012)
37. Wright, C.: On the coherence of vague predicates. Synthese. **30**, 325–365 (1975)
38. Woo, J.S., Kang, H.G.: Risk attitudes of investment bankers: are they risk-lovers? Experiment and survey on investment bankers. In: Conference of the Korean Financial Association, Bangkok, pp. 705–773 (2016)
39. Wu, D., Mendel, J.M., Joo, J.: Linguistic summarization using if-then rules. In: 2010 IEEE International Conference on Fuzzy Systems, Barcelona, pp. 1–8 (2010)
40. Yager, R.R.: A new approach to the summarization of data. Inf. Sci. **28**, 69–86 (1982)
41. Yager, R.R., Ford, K.M., Cañas, A.J.: An approach to the linguistic summarization of data. In: Bouchon-Meunier, B., Yager, R.R., Zadeh, L.A. (eds.) IPMU 1990. LNCS, vol. 521, pp. 456–468. Springer, Heidelberg (1991). https://doi.org/10.1007/BFb0028132
42. Zadeh, L.A.: From computing with numbers to computing with words - from manipulation of measurements to manipulation of perceptions. In: Wang, P. (ed.) Computing with Words, pp. 35–68. Wiley, New York (2001)
43. Zaleskiewicz, T., Piskorz, Z., Borkowska, A.: Fear or money? Decisions on insuring oneself against flood. Risk Decis. Policy **7**, 221–233 (2002)

Data Management and Quality

Advanced Data Integration with Signifiers: Case Studies for Rail Automation

Alexander Wurl[1(✉)], Andreas Falkner[1], Alois Haselböck[1], and Alexandra Mazak[2]

[1] Siemens AG Österreich, Corporate Technology, Vienna, Austria
{alexander.wurl,andreas.a.falkner,alois.haselboeck}@siemens.com
[2] Business Informatics Group, TU Wien, Vienna, Austria
mazak@big.tuwien.ac.at

Abstract. In Rail Automation, planning future projects requires the integration of business-critical data from heterogeneous, often noisy data sources. Current integration approaches often neglect uncertainties and inconsistencies in the integration process and thus cannot guarantee the necessary data quality. To tackle these issues, we propose a semi-automated process for data import, where the user resolves ambiguous data classifications. The task of finding the correct data warehouse entry for a source value in a proprietary, often semi-structured format is supported by the notion of a *signifier* which is a natural extension of composite primary keys. In three different case studies we show that this approach (i) facilitates high-quality data integration while minimizing user interaction, (ii) leverages approximate name matching of railway station and entity names, (iii) contributes to extract features from contextual data for data cross-checks and thus supports the planning phases of railway projects.

Keywords: Data integration · Signifier · Data quality

1 Introduction

In order to properly plan the utilization of production capacity, e.g., in a Rail Automation factory, information and data from all business processes and project phases must be taken into account. Sales people scan the market and estimate the number of assets (i.e., producible units) of various types (e.g., hardware control units for signals) which will be potentially ordered within the next years. Estimation is based on features such as various types of signals, points,

This work is funded by the Austrian Research Promotion Agency (FFG) under grant 852658 (CODA).
Alexandra Mazak is affiliated with the Christian Doppler Laboratory on Model-Integrated Smart Production at TU Wien.

J. Filipe et al. (Eds.): DATA 2017, CCIS 814, pp. 87–110, 2018.
https://doi.org/10.1007/978-3-319-94809-6_5

or crossings of the relevant railway stations. The numbers of assets get refined phase by phase, from bid preparation to order fulfillment, ending up in a detailed bill-of-materials containing the required asset variants, e.g., for specific signal and lamp types. Since these phases often are carried out by different departments with different requirements and interests, the same features and assets are described by different properties with - perhaps slightly - different contents and in different proprietary formats, like spreadsheets or XML files. In addition to company-internal data, contextual data such as open maps or weather data from the Internet can be used to derive and enrich information about necessary assets. Proprietary structures as well as heterogeneous feature and asset representations hinder the process of mapping and merging information which is crucial for a smooth overall process and for efficient data analytics that aims at optimizing future projects based upon experiences from all phases of previous projects. One solution approach is to use a data warehouse and to map all heterogeneous data sets of the different departments and process phases to a unified data schema.

To achieve high data quality, it is important to avoid uncertainties and inconsistencies while integrating data into the data warehouse as storing duplicate or contradicting information may have business-critical effects. Part of the information can be used to identify corresponding data (i.e., when used as key), part of it can be seen as relevant values such as quantities and costs. Only if keys of existing information objects in the data warehouse are comparable to those of newly added information from heterogeneous data sets, that information can be stored unambiguously and its values are referenced correctly.

Keys are formed from one or more components of the information object and are significant for comparing information of heterogeneous data sets with information stored in the data warehouse. There are two significantly different causes for the case that two of such object keys do not match: (i) the two objects represent the same real-world object and should therefore have the same key but their keys (slightly) differ from each other, and (ii) the two objects are to be considered as different and should have different keys. Using solely heuristic lexicographical algorithms [1] to automatically find proper matches does not necessarily succeed to reliably distinguish those two cases because each single character of the key might be important or not. Inappropriate heuristics lead to wrong matching results, which may have major consequences to business. Therefore it is critical to rely on semantics for matching algorithms.

To solve this problem, we adapt the concept of *signifiers* [2,3]. For each data set type in the data warehouse, a domain expert defines a signifier as a combination of identifying properties of the objects. To deal with approximate matches by variants of object names, each signifier component can have aliases. Integration of a new data set to the data warehouse is based on signifier matching to identify or classify objects. In case of mismatches between the data warehouse and the new data set during data integration, a manual control either confirms the mismatch or interacts aiming for a match. The signifier serves as a key to uniquely identify an object in the course of normalization of information in the data warehouse.

The remainder of this paper contains first a comprehensive problem definition in Sect. 2 where we reveal challenges of integrating heterogeneous data sets into a data warehouse in an industrial context. In Sect. 3, we specify our solution approach of using signifiers to avoid inconsistencies between assets in the data warehouse and corresponding quality metrics in a formal way. Sections 4, 5, and 6 describe the details of three case studies which we carried out in the Rail Automation domain to validate our approach. Finally, Sect. 7 discusses related work and Sect. 8 concludes the paper with a summary and an outlook.

2 Problem Definition

Basically, the degree of data quality in industrial environments such as Rail Automation depends on the exploitation of heterogeneous data sources, which are related to each other and together contribute to a higher information value. We aim at connecting related data entities - not only records in databases of structured railway data but also semi-structured data such as spreadsheets and text files used by the various involved departments. Depending on the purpose, the data entities are structured differently and deviate in naming conventions. Figure 1 illustrates the intent to connect planning, configuration, operational, and contextual data by integrating data into a data warehouse.

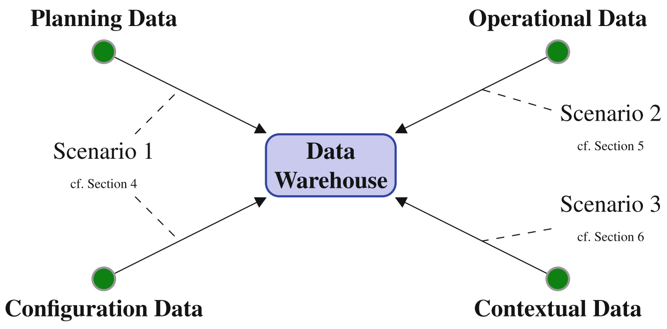


Fig. 1. Interplay of heterogeneous data sources in Rail Automation.

Planning Data. Planning data are necessary for bid proposal management and obsolescence management of assets. In the application domain of Rail Automation, data sets comprise quantity estimations of station features (like the number of signals and points, partly distinguished by types and subtypes) and corresponding assets that must be available at certain points in time.

Configuration Data. Data sets of configuration data comprise all engineering data representing a system in the field. Consequently, the corresponding numbers of installed assets ensure reliable data.

Operational Data. Operational data describe active and anomalous states of railway stations and assets installed in the field. Usually, such data are log files containing states of subsystems and communication between different subsystems, like telegrams between train and control system.

Contextual Data. In the context of this paper, we make use of the open data platform OpenRailwayMap[1], a queryable online map of the world's railway infrastructure based on OpenStreetMap[2].

Integrating these heterogeneous data sources, the following scenarios arise and will be tackled in respective case studies:

Scenario 1: Planning and Configuration Data. Both data sets comprise features and assets related to each other over the various phases and different projects. Since there is no standardized identification convention of entities in heterogeneous data sets, care must be taken not to wrongly store duplicates of the same information in the integration process to a data warehouse. Entities and their attributes require a matching technique in order to avoid false duplicates over different phases of a system and to allow proper aggregations (sums) of asset numbers over different projects.

Scenario 2: Operational Data. Whereas configuration data define the static assets of railway stations, involving operational data allows to improve maintenance by analyzing failure conditions of assets. For that, it is vital to match the descriptor of an entity in a log file to an object in an other data set or a given data warehouse. In the case study for this scenario we focus on station name matching, where different naming conventions of station names prohibit a look-up of a station name in a data source by a hash-based look-up or simple string comparison.

Scenario 3: Contextual Data. The integration of contextual data, like OpenRailwayData, is an additional source of information for enrichment or confirmation of data. For both cases, matching feature and asset names of contextual data sources with corresponding features and assets of company-internal data sets is necessary.

As indicated in Fig. 1, we deal with these data integration scenarios by mapping data into a conjoint data warehouse. The crucial task is the mapping, i.e., given an object of a data source, how can we identify an existing version of this object in the data warehouse. Figure 2 shows an example with two data sources: a spreadsheet (such as Microsoft Excel) and an XML file. Each row of the spreadsheet represents an information object, e.g., the expected quantity from a planning phase. "Source1" in Fig. 2 has five columns where the first three are used to compute a key (by function "extractKey1") and the fourth column (selected by function "extractValue1") contains a relevant value, e.g., the quantity in form of a numerical value.

Another format is XML which contains structured objects (such as representations of physical objects), e.g., a bill of materials from an installation phase. "SourceN" in Fig. 2 shows some objects. Each object consists of sub-elements in which all information of the corresponding object is included - either for use in keys (accessed by function "extractKeyN") or for values. In this example,

[1] http://www.openrailwaymap.org/.
[2] http://www.openstreetmap.org/.

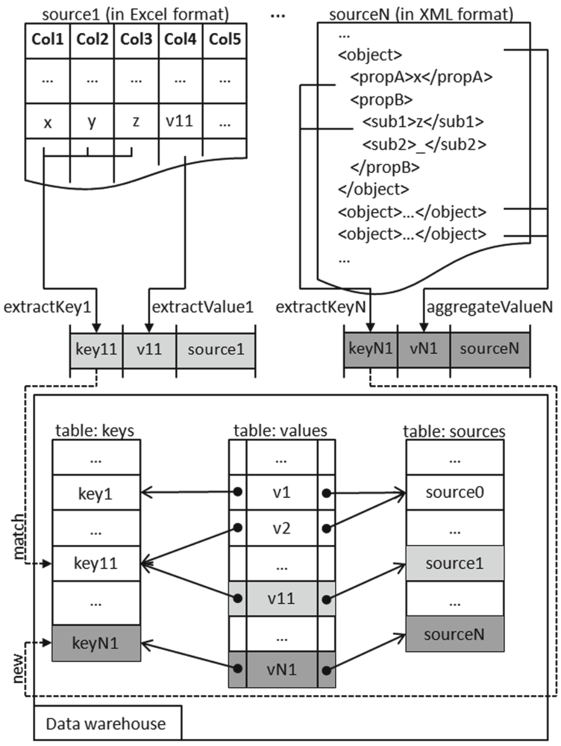


Fig. 2. Integration scenario of heterogeneous data sets. (Source: [3]).

the value is not selected directly, but aggregated from all objects with the same key (using aggregation function "aggregateValueN"), e.g., by counting all those object instances to derive their quantity as a numerical value.

The essence of each data source can be seen as a set of triples, containing a key, a value, and the source identifier. As an intermediate representation, those triples are merged into the data warehouse which comprises the history of all values with references to data sources and keys. The keys are necessary to map different data representations of the same information from different data sources onto each other. If a key from a triple matches an existing key, the latter is reused, e.g., "key11" for the triple from "source1" in Fig. 2. Else, a new key is added to the data warehouse, e.g., "keyN1" for the triple from "sourceN". This means that the new information does not correspond to other information in the data warehouse.

Such an integration scenario is reflected in the integration of heterogeneous data sources (cf. Fig. 1) and poses the following questions that will be addressed in the case studies below:

– Can a simple approach, such as to assemble some properties as components for a unique identification key, cover many use cases?

– High heterogeneity in technical aspects, data model, and semantics requires an advanced integration approach. How can extraction functions be defined in a systematic way?
– How shall a match be defined in detail? Perfect match vs. approximate match (case-sensitivity, lexicographical distance)? How can wrong matches be avoided?
– How to decide whether (syntactically) not matching keys refer to the same information? Are synonyms used?
– The process of comparing keys needs some user interaction from domain experts. What is the best process? How to minimize efforts?

3 Using Signifiers for Data Integration

We propose a strategic technique in the ETL-process to instantly react on potential inconsistencies, e.g., when there is not a perfect match of a source object with an object in the data warehouse. Instead of names or IDs, we use and extend the concept of *signifiers*, as introduced in [2], for mapping an object of a data source to the right object type in the data target (the data warehouse). A signifier consists of different components. To check if two objects match, the components of their signifiers are checked pairwise.

To cope with the usage of different wordings or words in different languages in the data source, we need two versions of signifiers: a source signifier and a target signifier which is extended by aliases.

Definition 1. *A **Source Signifier** is an n-tuple of strings. The term S_i refers to the i^{th} component of a source signifier S.*

The meaning of each element of the signifier is determined by its position in the tuple. In the railway asset management example described in Sect. 2, we use signifiers of length 3, representing category, subcategory and subsubcategory, respectively. Example: (“Signal”, “Main Signal”, “8 Lamps”).

Definition 2. *A **Target Signifier** is an n-tuple of sets of strings. The term T_i refers to the i^{th} component of a target signifier T.*

Target signifiers allow to specify more than one string per component. These strings represent aliases. Example: ({“Signal”, “S”}, {“Main Signal”, “MainSig”, “Hauptsignal”, “HS”}, {“8”, “8 Lamps”, “8lamps”}).

The main task of integrating a new object from a data source into a target data warehouse is to match source and target signifiers. To be able to deal with approximate matches, too, we use a distance function $dist(s, t)$, returning a value from $[0, 1]$, where 0 means that the two strings s and t are equal. There are many well-studied string metrics that can be used here – see, e.g., [1]. Given a string distance function $dist(.,.)$, we define the minimum distance of a string s and a set of strings ts by: $dist_{min}(s, ts) = \min_{i=1..|ts|} dist(s, ts_i)$.

In order to express different significances of different components of a signifier, we use weight factors w_i for each component i. The total sum of all weighted components of a signifier is 1. In the case study in Sect. 4, the weighting of the components is demonstrated.

Definition 3. *Let S be a source signifier and T be a target signifier with n components. Let dist(.,.) be a string distance function. Let w_i be component weights. The function **Dist(S,T)** returns a value from* $[0, 1]$ *and is defined in the following way:*

$$Dist(S,T) = \sum_{i=1..n} dist_{min}(S_i, T_i) * w_i \tag{1}$$

Now we are in the position to formally define perfect and approximate matches. In the following definitions, let S be a source signifier and TS be a set of target signifiers.

Definition 4. *Let $\tau_1 \in [0, 1]$ be the perfect match threshold value, usually the value 0 or close to 0. The signifier $T \in TS$ is a **perfect match**, if and only if $Dist(S,T) \leq \tau_1$ and there is no other $T' \in TS$ ($T' \neq T$) which is also a perfect match.*

Definition 5. *Let $\tau_2 \in [0, 1]$ be the approximate match threshold value, usually a small value like 0.20. The signifier $T \in TS$ is an **approximate match**, if and only if $Dist(S,T) \leq \tau_2$ and T is not a perfect match.*

The algorithm of integrating a source data object into the target database (data warehouse) is a semi-automated task based on the previously defined perfect and approximate matches. If a perfect match is found, the new object is automatically assigned to the target data warehouse. In all other cases, the user is asked to decide what to do. The following steps summarize this algorithm for adding a source object with signifier S to a target database with existing target signifiers TS:

1. If we find a $T \in TS$ that is a perfect match to S, add the new object to the target database using T. The aliases of T are updated in the target database by adding all components of S, that do not fully match by string compare, to the aliases in T. Done.
2. Let TS_{approx} be the (possibly empty) set of target signifiers that are approximate matches to S. Ask the user for a decision with the following options:
   (a) Accept one $T \in TS_{approx}$ as match. The new object will be added to the target database using T. The aliases of T are updated in the target database by adding all components of S, that do not fully match by string compare, to the aliases in T.
   (b) Accept S as a new target signifier. The new object will be added to the database and S will be added to the target signifiers.
   (c) Abort import process and fix the source database.
3. For all other signifiers S (non-perfect and non-approximate), ask the user for a decision with the following options:
   (a) Accept one $T \in TS$ as match.

(b) Accept S as a new target signifier.
(c) Abort import process and fix the source database.

Each import of a data source potentially increases and completes the number of target signifiers or accordingly the aliases of target signifiers. By that, user interaction will decrease over time and the import of source data will get closer and closer to run fully automatically.

In order to compare different signifier parameter settings (like threshold values or string distance functions), we define the following performance indicator.

Definition 6. *Let SS be a set of source signifiers. Let TS be a set of target signifiers. Let SS be partitioned into the following subsets:* $SS_{perfect}$ *(source signifiers that have a perfect match in TS),* SS_{approx} *(source signifiers that have approximate matches in TS),* $SS_{nonmatch}$ *(the rest). The* ***Signifier Performance Indicator*** *consists of the following components:* $p/a(\mu)/n$ *that are defined as follows:*

$p = \frac{|SS_{perfect}|}{|SS|}$, *the portion of perfect matches*
$a = \frac{|SS_{approx}|}{|SS|}$, *the portion of approximate matches*
μ *is the average size of the sets of approximate matches*
$n = \frac{|SS_{nonmatch}|}{|SS|}$, *the portion of non-matches*

A "good" signifier performance indicator aims for high p values and low a, μ and n values. A high p value means: many source objects could be identified in the target database safely without user interaction. The a value is the fraction of source objects for which the user had to select from a set of options (the approximate matches). The average size of these sets of options is indicated by μ. For the fraction n of source objects, user interaction is even more expensive, because no set of likely matching objects could be presented by the tool. Clearly, $p + a + n = 1$. For example, a source signifier set size of 100 and a performance indicator 0.5/0.4(3.5)/0.1 means: 50 objects matched automatically, for 40 objects the user selected from a list with average size of 3.5, and for 10 cases the user had to decide without tool support.

4 Case Study 1: Signifiers in the Integration Process

Our empirical case studies are based on the guidelines introduced in [4]. The main goal is to evaluate if the approach using signifiers for the integration of data from heterogeneous data sets into the data warehouse significantly improves data quality, i.e., reduces incorrect classifications of objects imported into a given data warehouse. Especially in the domain of Rail Automation with business critical data it is crucial to avoid incorrect classification of data. We conducted the case study for the business unit Rail Automation, in particular for importing planning, order calculation and configuration data of railway systems into an asset management repository.

4.1 Research Questions

Q1. Can signifiers, as described in Sect. 3, minimize the incorrect classification of objects at data import?

Q2. Can our import process based on signifiers minimize user interactions?

Q3. From an implementation point of view, how easily can conventional (composite) primary keys of objects in a data warehouse be extended to signifiers?

4.2 Case Study Design

Requirements. We perform our empirical tests in an asset management scenario for Rail Automation. A data warehouse should collect the amount of all assets (i.e., hardware modules and devices) of all projects and stations that are installed, currently engineered, or planned for future projects. To populate this data warehouse, available documents comprising planned and installed systems should be imported. Planning and proposal data, available in Excel format, can be classified as semi-structured input: While the column structuring of the tables is quite stable, the names of the different assets often differ because of the usage of different wording, languages, and formatting. The tables also contain typing errors, because they are filled out manually as plain text. Configuration data for already installed systems is available in XML format and is therefore well-structured. As the underlying XML schema changed over time due to different engineering tool versions, also XML data contain structural variability.

The exemplary import of a couple of different Excel files, including files from projects of different countries, and one XML file should be performed. The test should start with an empty data warehouse, i.e., no target signifiers are known at the beginning; the set of appropriate target signifiers should be built up during import of the different files.

Setup. The software architecture for our implementation of the ETL process for the case study is sketched in Fig. 3. We use a standard ETL process as, e.g., described in [5]. The extraction phase is implemented with KNIME[3].

The result of the extraction phase is a dataset containing source signifiers and corresponding data values. Our Signifier Matching component, implemented in Python, tries to find for each entry in that dataset a matching target signifier already stored in the data warehouse. Ambiguities are resolved by asking the user. The load phase consists of updates to the data warehouse: input data are added with their references to a target signifier and a source descriptor. New target signifiers and new aliases of existing target signifiers are added, as well.

Using our method, the following application-specific parameters must be adjusted: the number of signifier components, the string distance metrics, the weights of the signifier components, and the threshold values for perfect and approximate matches. In our tests, we used signifiers consisting of 3 components, representing category, subcategory and subsubcategory of a data object. For

[3] https://www.knime.org/.

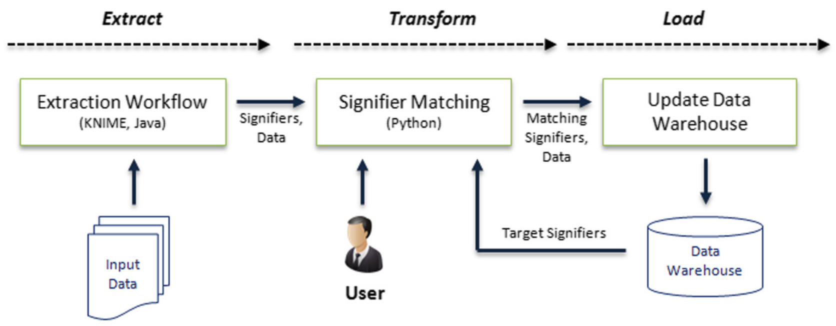


Fig. 3. Case study ETL process. (Source: [3]).

string comparison we used a case-insensitive Jaro-Winkler distance [1]. Weights and threshold are determined as shown in the next section, specifically in Table 4.

4.3 Results

In this section, we present the results of our case study from data and behavioral perspectives. Our main goal was to analyze the applicability of signifiers in the transform phase of the ETL process with respect to data quality and amount of necessary user decisions.

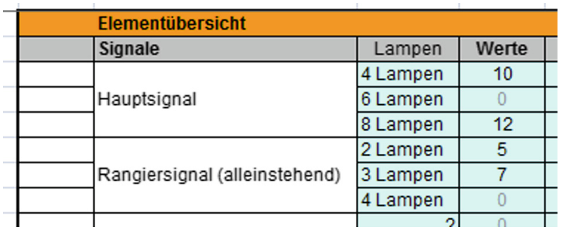
Data Integration with Signifiers. We demonstrate our approach by describing the key steps of data integration of an Excel source and an XML source on a concrete example from the Rail Automation domain. We are sketching the main database tables according to Fig. 2.

Target signifiers are represented in a table as depicted in Table 1. Each row represents an asset type with a primary key, ID, for being referenced from data tables. The other columns represent the components of the target signifiers. Please note that such components typically contain not only a single string but a set of strings (i.e., aliases) for different wordings or different languages.

Table 1. Target signifier table of the data warehouse. (Source: [3]).

ID	Category	SubCat.	SubSubCat.
1	Signal	Main Signal, Hauptsignal, HS	4 lamps, 4
2	Signal	Shunting Signal, Rangiersignal, RS	2 lamps, 2
...	...	...	...

Figure 4 shows a small part of an input table (in German language). The extraction of the first row results in the source signifier `[Signale, Hauptsignal, 4 Lampen]` and value 10 representing the amount of main signals with 4 lamps in the railway station. Obviously, this signifier has no perfect match in the target signifiers as shown in Table 1, because German language and some extended wording is used in the source Excel file. The matching distances according to our settings are 0.05 for the first and 0.182 for the second target signifier. The first one has a clearly lower distance value. After the user has confirmed that this is the right match, firstly, the target signifier table is updated by adding aliases “Signale” (for category) and “4 Lampen” (for subsubcategory) to signifier 1, and secondly, the object value 10 is added to the data table (see first row in Table 2) with links to the right target signifier and its source. The other rows are extracted in the same way (not shown in the table).



	Elementübersicht		
	Signale	Lampen	Werte
	Hauptsignal	4 Lampen	10
		6 Lampen	0
		8 Lampen	12
	Rangiersignal (alleinstehend)	2 Lampen	5
		3 Lampen	7
		4 Lampen	0
		2	0

Fig. 4. An excerpt of objects in an Excel spreadsheet. (Source: [3]).

For the XML source document depicted in Listing 1, the signifier components for the first object are built in the following way: The first component is the tag name of the XML element, `signal`, the second one its signal type, `HS`, and the third one is created by counting the lamp sub-elements. Now we get the source signifier `[signal, HS, 4]`. Since our distant metrics are case-insensitive, we have a perfect match with target signifier 1 in the data warehouse. The data value for this kind of object is computed by counting the number of objects of this type in the XML file. After processing the second XML element in a similar way, the data table is updated. Table 2 shows the resulting two entries at the end.

```
<?xml version=”1.0” encoding=”UTF−8”?>
<DATA−FORMAT schema−version=”23”>
        <signal id=”4463”>
                <Name type=”string” state=”30”>A</Name>
                <signaltype type=”string” state=”40”>HS</signaltype>
                ...
                <lamps type=”assoc” state=”30”>
                        <_REF>4362</_REF>
                        <_REF>4361</_REF>
                        <_REF>4363</_REF>
                        <_REF>4364</_REF>
                </lamps>
        </signal>
        <signal id=”4462”>
                <Name type=”string” state=”30”>R</Name>
                <signaltype type=”string” state=”40”>RS</signaltype>
                ...
                <lamps type=”assoc” state=”30”>
                        <_REF>4112</_REF>
                        <_REF>4113</_REF>
                </lamps>
        </signal>
</DATA−FORMAT>
```

Listing 1. An XML data source containing object elements. (Source: [3]).

Table 2. Data objects table after import of values from an Excel and an XML source. (Source: [3]).

Signifier	Value	Source
1	10	Source 1 (Excel)
...	...	...
1	1	Source 2 (XML)
2	1	Source 2 (XML)

Results in Data Quality and User Interactions. We performed our tests on several data sources in Excel and XML format, starting with an empty data warehouse without any target signifiers. In total about 90 signifiers were created; 10% of these signifiers got aliases. The different sources partly contained different wordings, languages, and abbreviations for object designation, and contained typing errors. Qualitatively speaking, for the most of these different wordings, our signifier matcher found approximate matches and could very specifically ask the user for a match decision. In case of no perfect match could be found, the number of provided options was mostly in the range of 3. A few signifiers had name variations which fell out of the class of approximate matches, and the user had to match them manually. Provided that incorrect object classifications should be avoided, the number of user decisions was minimal.

Table 3. The correlation of match candidates and expert reference (correct match).

	Correct match	No
Match candidate	tp	fp
No	fn	tn

For a quantitative assessment of the quality of matches we use the F-measure (more specifically, the F_2-measure) from the field of information retrieval [6,7]. This measure is based on the notion of *precision* and *recall*, which are defined in terms of true/false positives/negatives. Table 3 shows how true/false positives/negatives are defined by comparing the correct value as defined by an expert with the match candidates computed by signifier matching (perfect or approximate matches). Precision, recall and the F_2-measure are defined as follows:

$$Precision = \frac{|tp|}{|tp| + |fp|} \tag{2}$$

$$Recall = \frac{|tp|}{|tp| + |fn|} \tag{3}$$

$$F_2 = 5 \times \frac{Precision \times Recall}{4 \times Precision + Recall} \tag{4}$$

Table 4. Precision, recall and F_2 values for signifier matching tests varying approximate match threshold (τ_2) and signifier component weights (w).

	$\tau_{21} = 0.00$	$\tau_{22} = 0.025$	$\tau_{23} = 0.05$	$\tau_{24} = 0.1$
w_1	P: 1.000	P: 0.891	P: 0.779	P: 0.710
	R: 0.891	R: 0.950	R: 0.956	R: 0.961
	F_2: 0.914	F_2: 0.938	F_2: 0.914	F_2: 0.898
w_2	P: 1.000	P: 0.930	P: 0.887	P: 0.835
	R: 0.891	R: 0.950	R: 0.950	R: 0.950
	F_2: 0.914	F_2: **0.946**	F_2: 0.937	F_2: 0.925
w_3	P: 1.000	P: 0.901	P: 0.793	P: 0.432
	R: 0.891	R: 0.950	R: 0.950	R: 0.956
	F_2: 0.914	F_2: 0.940	F_2: 0.914	F_2: 0.770

Precision is the number of correct match candidates divided by the number of all match candidates. Recall is the number of correct match candidates divided by the number of matches that should have been returned (i.e., all correct matches). The F-measure combines these two values. We use the F_2-measure here to emphasize the recall value; it is important in our application that we do not miss any correct matches.

Table 4 shows the results of our tests by precision, recall and F_2 values. We set the perfect match threshold τ_1 to 0. We varied the approximate match threshold τ_2 and weight values to find an optimal combination. We used threshold values $\tau_{21} = 0.01$, $\tau_{22} = 0.025$, $\tau_{23} = 0.05$, $\tau_{24} = 0.1$; we used weight values $w_1 = (\frac{1}{14}, \frac{4}{14}, \frac{9}{14})$, $w_2 = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, $w_3 = (\frac{9}{14}, \frac{4}{14}, \frac{1}{14})$. It was observed that:

- As expected, small threshold values lead to high precision values, large threshold values lead to high recall values. Precision and recall are negatively correlated.
- For the set of used test data a value combination $\tau_{22} = 0.025$ and $w_2 = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ achieves the best matching results, i.e., has the highest F_2-measure value.
- The F_2 value of many threshold/weight combinations are - in absolute terms - quite high, indicating that the proposed method achieves high-quality matches and is quite robust against small changes of input parameters.

For a quantitative assessment of the user interaction performance we use the Signifier Performance Indicator as defined in Sect. 3. Table 5 shows the results for our test scenario. The perfect match fraction is equal in all settings (about 72%) because always $\tau_1 = 0$. Naturally, the approximate match fraction grows with larger τ_2. The average size of the approximate matches is very small, making it easy for the user to make a decision. The fraction of non-matching cases is reasonable small, provided that we started with an empty data warehouse so that the first set of source data is always non-matching. We would have expected that w_1 with its focus on the last key component would perform well as the last component is very discriminative (already one different digit makes a big difference, e.g. for lamp numbers in signal types). However, quite to the contrary, the top-down based w_3 with a focus on the first component performs better - most probably due to the wide variance of different entity types (not just signals).

The bold values in the tables show that different thresholds and weight strategies perform better for quality metrics (F_2 measure in Table 4) and interaction (Signifier Performance Indicator in Table 5). Users need to decide what is more important to them.

4.4 Interpretation of Results

We analyze the results with regard to our research questions.

Q1. *Can signifiers minimize the incorrect classification of objects?* Yes, according to the test results shown in Table 4, our method is capable to achieve high values of precision and recall values. Compared to the "perfect match only" scenario ($\tau = 0$) with its perfect precision, approximate matches showed a weaker precision but a much better recall and therefore a better F-measure.

Q2. *Can our import process based on signifiers minimize user interactions?* Yes, provided that automated matches should only be made based on a perfect match, the number of user interactions were minimal in the sense that the user was not asked for a similar match twice. Furthermore, the list presented to the user for a manual match was sorted by match distance; in most cases the user found his/her match on the first or second place in that list. These observations are quantitatively confirmed by the corresponding signifier performance indicators as shown in Table 5.

Q3. *From an implementation point of view, how easily can conventional (composite) primary keys of objects in a data warehouse be extended to signifiers?* Instead

Table 5. Signifier Performance Indicators for signifier matching tests varying approximate match threshold (τ_2) and signifier component weights (w).

	$\tau_{21} = 0.00$	$\tau_{22} = 0.025$	$\tau_{23} = 0.05$	$\tau_{24} = 0.1$
w_1	p: 0.716	p: 0.716	p: 0.716	p: 0.716
	$a(\mu)$: 0.000(0.0)	$a(\mu)$: 0.072(1.4)	$a(\mu)$: 0.101(1.8)	$a(\mu)$: 0.125(2.0)
	n: 0.284	n: 0.212	n: 0.183	n: 0.159
w_2	p: 0.716	p: 0.716	p: 0.716	p: 0.716
	$a(\mu)$: 0.000(0.0)	$a(\mu)$: 0.056(1.3)	$a(\mu)$: 0.076(1.4)	$a(\mu)$: 0.083(1.5)
	n: 0.284	n: 0.228	n: 0.208	n: 0.201
w_3	p: 0.716	p: 0.716	p: 0.716	p: **0.716**
	$a(\mu)$: 0.000(0.0)	$a(\mu)$: 0.081(1.2)	$a(\mu)$: 0.106(1.7)	$a(\mu)$: **0.179(3.6)**
	n: 0.284	n: 0.203	n: 0.178	n: **0.105**

of using data tables with composite keys in the data warehouse, signifiers are stored in a separate table with a generated key for referencing from data tables. Design and implementation of this separate signifier table is straight-forward.

4.5 Threats to Validity

The first import of a data source where all signifier components were translated to another language produces a lot of user interaction. This could be avoided by generating and adding translations as aliases to the signifiers in the data warehouse.

In situations where a signifier component represents a number (e.g., number of lamps of a railway signal), plain string distance is not an optimal choice. E.g., a human would consider “2” closer to “3” than to “5”, which is usually not the case in a string distance function. Our definition of signifiers as a tuple of string components could be extended to components of different types. This would allow the implementation of type-specific distance functions and solve that problem.

5 Case Study 2: Railway Station Name Matching

An important subproblem of data exchange/integration tasks in the railway domain is the identification of a railway station by a name. Station names are not standardized; companies or even organization units often uses a slightly different diction and convention for station names. A typical example is the naming of the railway station of the small Austrian city Hainburg: “Hainburg”, “Hainburg a.d. Donau”, “Hainburg an der Donau”, “Bf. Hainburg a.d. Donau”, “Hainburg Hst.”, ...

In this section, we demonstrate the advantages of using signifiers on the railway station look-up problem: Given is a database table R containing data for railway stations. R has at least one, but typically a couple of columns representing station names. The look-up task is to find the right railway station record in R identified by a given string S, where S not necessarily perfectly match (by string compare) a name field in R. Either the right railway station could be identified in R with high reliability, or a human expert has to choose from a (small) set of approximate matches provided by the tool.

5.1 Research Questions

Q1. Can signifiers, as described in Sect. 3, be used to contribute to a solution to the railway station look-up problem?

Q2. On looking up many railway station names, what is the performance in terms of user interactions?

Q3. On looking up many railway station names, what is the performance in terms of classification correctness?

5.2 Case Study Design

For our evaluation we use the following existing railway station databases:

R1. A table with descriptions of all railway stations in Austria, provided by the Austrian Federal Railways. This table contains 3 columns with partly different railway station names. The size is about 5200 records.

R2. Railway station data from the OpenRailwayMap. This table contains different columns of alternative station names, like "name", "uic_name", "wikipedia". We queried for Austrian railway stations and extracted about 1300 records.

From operation data we collected the list RO of ca. 700 railway station names. These names were defined by the responsible configuration engineers and often do not perfectly match any station names of R1 and R2. The goal is to find the right records in R1 and R2 for all the station names of RO. We are using the framework of source and target signifiers as defined in Sect. 3 for station name matching. We use the following signifier settings: number of components = 1, trivially with weight 1.0; perfect match threshold $\tau_1 = 0.05$ and approximate match threshold $\tau_2 = 0.2$; string distance function: Jaro-Winkler. The specific setting of the thresholds has been experimentally determined.

We observed that the standard string distance function (Jaro-Winkler in our case) could easily be specialized by language- and domain-specific adjustments to considerably improve matching quality. When doing a string distance calculation $Dist(a, b)$, we used the following string pre-processing steps on the strings a and b:

M1. Non-alphanumeric characters are removed from a and b. The strings a and b are converted to lowercase.

M2. M1 plus language-specific modifications: German umlauts are converted, like 'ö' to 'oe'.

M3. M2 plus domain-specific stopwords are removed, like "Bahnhof" (German for station), "Bhf.", etc.

5.3 Results

In Fig. 5, the semi-interactive process of looking up station names from RO in database tables R1 or R2. Names with a perfect match signifier distance are considered to be found without any user interaction. Approximate and non-matches require user interaction.

Figure 6 shows the results of our experimental matching of all strings in the configuration data set RO against the naming strings of the databases R1 and R2. The three curves shown in the figure correspond to the string distance strategies M1, M2, and M3.

The legends in Fig. 6 show the signifier performance indicators for the corresponding settings.

The correctness of the look-ups is best described by the precision value $\frac{|tp|}{|tp|+|fp|}$. We count as true positives tp the correct perfect matches plus - in

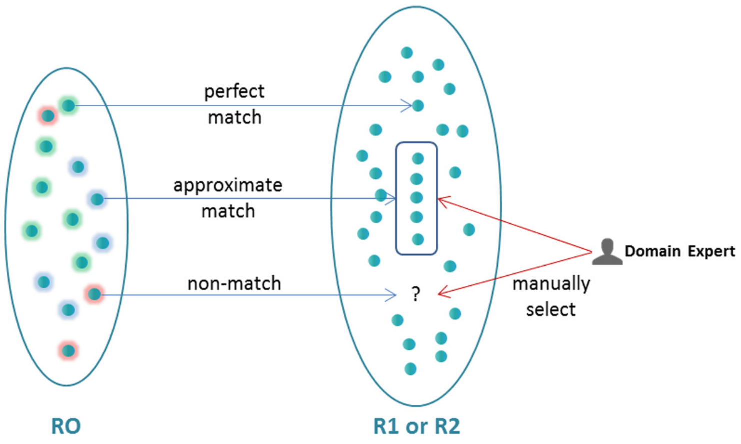


Fig. 5. Semi-interactive process of looking up source signifier names from RO in data sets R1 or R2.

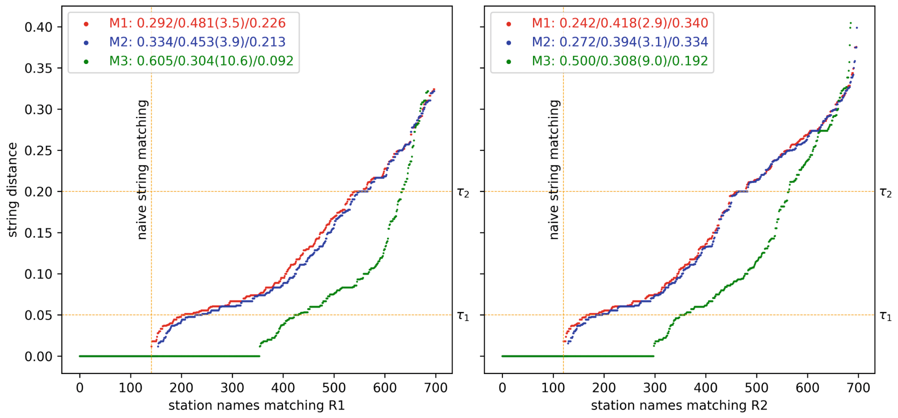


Fig. 6. Results for looking-up station names from configuration data RO in data sets R1 and R2 according to the 3 different string matching functions M1, M2, and M3.

case no perfect match could be found - those cases where the correct value was in the set of the approximate matches. The rest of the cases are counted as false positives fp. A manual review of the resulting look-ups showed that in only 2 cases, i.e. 0.3%, the perfect match was wrong. The overall number of false positives was 6.4%, resulting in a precision value of 93,6%.

5.4 Interpretation of Results

We analyze the results with regard to our research questions.

Q1. *Can signifiers be used to contribute to a solution to the railway station look-up problem?* With a few experiments on signifier settings (like threshold values) and some simple language- and domain-specific tweaks of the string distance function, approximate name matching based on signifiers (see Fig. 5) is a robust and flexible framework for dealing with the different naming conventions of railway station data and therefore for solving the station look-up problem.

Q2. *What is the performance in terms of user interactions?* There is a considerable improvement of automated and approximate matches from string matching strategy M1 to M3. Compared to a naive string matching approach indicated by the vertical line in Fig. 6, M3 and our method based on signifier matching shows good performance: More than 60% (R1) resp. 50% (R2) could be matched automatically. About 30% are approximate matches and only about 10% (R1) resp. 20% (R2) could not be matched by signifiers at all.

Q3. *What is the performance in terms of classification correctness?* Considering high automation and therefore user interaction minimization, the measured precision value of 93,6% is acceptable.

5.5 Threats to Validity

The right, application-specific setting of the signifier parameters is important. If the perfect and approximate threshold values τ_1 and τ_2 are too small, automation performance will be compromised. If they are too large, chances of wrong matches rise. Therefore a balanced tuning of these parameters is crucial. Similarly, the choice of the string distance function is of high importance. Actually, adaptations of standard approaches - like the removal of stopwords - was necessary as shown above. Experiments with other string matching strategies, like phonetic matching [8] or fuzzy matching [9] did not produce better results.

6 Case Study 3: Automated Feature Extraction from Open Data Sources

In the bid phase of railway projects, the engineering team has to determine the quantities of required assets as a crucial contribution to the estimated overall costs of the system to be offered. There are several types of features of a railway project that allow an estimation of the number and types of assets with reasonable precision. These are, among others, the number of railway signals, switches, and level crossings. Estimation is usually done by counting (in plans or in the field) and/or by comparison with past projects of similar size. The first method is time consuming, the second one is usually not precise enough. An alternative, or at least a complementary way of determining these needed features is to automatically extract them from open, contextual data sources, like OpenRailwayMap. This approach has the potential to speed up the cost estimation process and to improve the reliability of the figures.

As described in the station name matching case study in Sect. 5, signifiers are used for finding the right station in the OpenRailwayMap by approximate station name matching. In this case study, we show how important features could automatically be extracted from this open data source. We demonstrate this by a concrete example - the retrieval of the number of main signals of an exemplary railway station.

6.1 Research Question

How can open data contribute to quantitative feature estimations? Which process can be used to identify and extract key features from open data sources?

6.2 Case Study Design

For our process of extracting feature quantities for railway lines and stations we use the open online map of world's railway infrastructure, OpenRailwayMap. The web-based data mining tool Overpass-Turbo[4] enables to retrieve data via queries through the Overpass API[5]. According to the Overpass API, the Overpass Query Language[6] provides to specify queries with statements. The output can be in the format of a map or of a query language result format, like JSON or CSV.

First, we find the right station in the data by station name matching. We used the city Amstetten for our experiments. Then we geographically enframe the area of the train station on an integrated map. This step requires some degree of exactness - a too large frame will probably lead to counting elements from neighboring stations, a too narrow frame will miss some elements. In the last step, we query the quantity of elements (in our case, main signals) in the area defined by our frame.

6.3 Results

Figure 7 illustrates the defined frame of the train station Amstetten in Austria including the main signals that belong to this area. Listing 2 shows a simple query of retrieving the number of main signals. We first define the output format, CSV, and properties of nodes to be queried. In line 2, we specify a node as the geographically defined area, i.e., the train station Amstetten, and specify the query for elements of type `railway:signal:main` (main signals) in this area. This step is only possible in cooperation with a domain expert, because the exact extent of this area determines what is part of the according station and what not. Line 3 outputs all main signal nodes within the defined area. These nodes include further information, e.g., GPS-coordinates, which serve as additional information in data integration. The names of the signals could be used as source signifiers for matches of signals in other data sets, like configuration or operation data. Finally, line 4 of our code clip simply prints the number of main signals - 61 signals in our case.

[4] https://overpass-turbo.eu/.
[5] http://www.overpass-api.de/.
[6] http://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL.

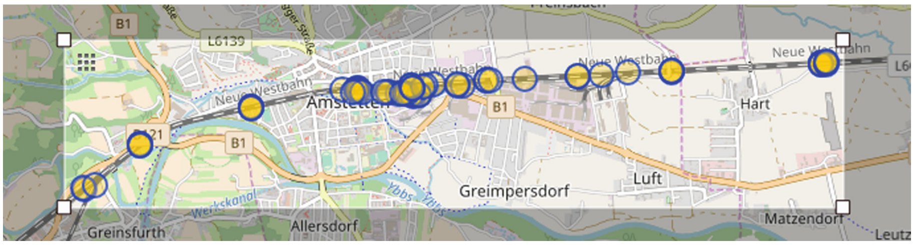


Fig. 7. Main signals of train station Amstetten, Austria.

In the final step of this scenario, we cross-check our results with company-internal integrated data. Since we could frame the geographical data in Open-RailwayMap in cooperation with a domain expert, the results match exactly, i.e., open data confirms the quantity of main signal in the train station Amstetten.

```
[out:csv(::id,::type,::count)];
node (48.1111, 14.83824, 48.12588, 14.93969)["railway:signal:main"];
out; //prints all nodes of main signals
out count; //result is 61 main signals
```

Listing 2. Counting main signals in the OpenRailwayMap area shown in Fig. 7.

6.4 Interpretation of Results

As shown in this case study, the process of deriving quantitative feature estimations from open data sets is simple and does not require high tool efforts and experiences. The results - in combination with approximate object matching based on signifiers - could be used for cross-checks and feature extraction automation. Cross-checking existing feature quantities enables continuous quality checks and optimizations of existing projects, particularly when these data are part of a data warehouse. Automated feature extraction could safe valuable time in the bid phase. The results are obtained with less effort and, if done as a diverse method, higher quality.

6.5 Threats to Validity

Accuracy of results crucially depends on the correct selection of the geographical area from where features are to be extracted. In addition, the object classification in the open data set (like `railway:signal:main`) must match the classification of the application. Different interpretations may distort the results.

7 Related Work

Data cleansing, also known as data cleaning, is an inevitable prerequisite to achieve data quality in the course of an ETL-process [10]. Naumann describes data cleansing as use case of data profiling to detect and monitor inconsistencies [5]. Resolving inconsistencies as part of the transformation phase has been a topic for the last two decades [11,12].

In the work of [5,13] tasks for ensuring data quality are classified; various data cleaning techniques have been proposed such as rule-based [14,15], outlier detection [16,17], missing values [18], and duplicate detection [19,20]. Most of these techniques require human involvement.

The work of [21,22] points out that integrating data is an iterative process with user interaction. Various approaches take this into consideration. Frameworks proposed in [23,24] enable the user to edit rules, master data, and to confirm the calculations leading to correct cleaning results. A higher detection accuracy in duplicate detection by a hybrid human-machine approach is achieved in the work of [20]. As presented in [18], numerous techniques are used to associate the data to get useful knowledge for data repairing, e.g., calculating similarities of contextual and linguistic matches being able to determine relationships. In [25] a logistic regression classifier learns from past repair preferences and predicts the type of repair needed to resolve an inconsistency.

As [26] reports that - although there are various data profiling tools to improve data quality - if people use them without having in mind a clear quality measurement method according to their needs, they are challenged by limited performance and by unexpectedly weak robustness. [27] claims that various frameworks offer the integration of heterogeneous data sets but a framework for quality issues such as naming conflicts, structural conflicts, missing values, changing dimensions has not been implemented in a tool at one place yet.

The work of [28] analyzes the representation of real-world objects in the context of ontology-driven situations. Similar to how real-world objects are characterized by attributes, in [2] the characteristics of models are described by a *signifier*. Basically, the concept of a signifier has its origin in the domain of model-driven engineering (MDE) where a signifier enhances the versioning system by describing the combination of features of model element types that convey the superior meaning of its instances. A signifier improves versioning phases in comparing and merging models leading to a higher quality of finally merged models. As we integrate objects from different sources, in our previous contribution [3] a signifier structures the combination of properties and finally improves data integration when objects to be integrated are compared and merged with objects in the data warehouse.

In the course of data mining, the work of [29] addresses real-world entities with blocking approaches based on schema-agnostic and schema-based configurations. Similarly to signifiers, a schema-based approach requires domain experts to manually choose blocking keys, i.e., splitting the record into categories to distinguish attributes pairwise. Traditionally, such requirements restrict the utility of schema-based approaches to be applied in other domains [30]. Using signifiers, in the domain of Rail Automation the blocks need to be predefined since we aim for explicit classifications of record matches. This means, unlike data mining approaches which seek for optimal combinations of blocking keys [31,32], signifiers serve to index records from heterogeneous data sources with noisy unstructured data. To prevent application restrictions in other domains, besides a defined matching threshold, blocks are weighted enabling to put domain-specific emphasize on blocks, e.g., matching with the first block is more relevant than the others.

8 Conclusion and Future Work

In this paper, we identified various issues in the integration process of railway data from heterogeneous, often noisy data sources, namely configuration, planning, operation, and contextual data. To address these issues, we proposed a semi-interactive approach. We introduced a technique using the notion of a *signifier* which is a natural extension of composite primary keys to support the user resolving ambiguous data classification. We validated the applicability of our approach in three cases studies.

In the first case study, we show that the approach using signifiers for the integration of data from heterogeneous data sets into the data warehouse has the potential to significantly improve data quality as well as to minimize user interactions.

In the second case study, we demonstrate the advantages of using signifiers on the railway station look-up problem, i.e., finding the right railway station in a dataset by approximate signifier matching strategies. An experimental evaluation verifies that signifiers leverage the performance of user interaction and classification correctness.

In the third case study, we show a beneficial effect in using signifiers for quantitative asset estimations. A concrete example demonstrates how querying contextual data (OpenRailwayMap) to extract railway station features can be used for cross-checks and can safe valuable time in the bid phase.

There are several ideas for future work. One relates to extending the textual representation of component types with numerical values. This affects the storage of values in the data warehouse as well as the algorithm that compares values. Another direction is to strive for a joint dictionary bridging language-specific component terms. This accelerates the integration process especially in companies with international respect.

References

1. Cohen, W.W., Ravikumar, P., Fienberg, S.E.: A comparison of string distance metrics for name-matching tasks. In: Proceedings of IJCAI-2003, 9–10 August 2003, Acapulco, Mexico, pp. 73–78 (2003)
2. Langer, P., Wimmer, M., Gray, J., Kappel, G., Vallecillo, A.: Language-specific model versioning based on signifiers. J. Object Technol. **11**, 4-1 (2012)
3. Wurl, A., Falkner, A., Haselböck, A., Mazak, A.: Using signifiers for data integration in rail automation. In: Proceedings of the 6th International Conference on Data Science, Technology and Applications - Volume 1: DATA, INSTICC, pp. 172–179. SciTePress (2017)

4. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Softw. Eng. **14**, 131 (2009)
5. Naumann, F.: Data profiling revisited. ACM SIGMOD Rec. **42**, 40–49 (2014)
6. Salton, G., Harman, D.: Information Retrieval. Wiley, Chichester (2003)
7. Wimmer, M., Langer, P.: A benchmark for model matching systems: the heterogeneous metamodel case. Softwaretechnik-Trends 33 (2013)
8. Ji, S., Li, G., Li, C., Feng, J.: Efficient interactive fuzzy keyword search. In: Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, 20–24 April 2009, pp. 371–380 (2009)
9. Zobel, J., Dart, P.W.: Phonetic string matching: lessons from information retrieval. In: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1996, 18–22 August 1996, Zurich, Switzerland, pp. 166–172 (1996). (Special Issue of the SIGIR Forum)
10. Bleiholder, J., Naumann, F.: Data fusion. ACM Comput. Surv. (CSUR) **41**, 1 (2009)
11. Leser, U., Naumann, F.: Informationsintegration - Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen. dpunkt.verlag (2007)
12. Sharma, S., Jain, R.: Modeling ETL process for data warehouse: an exploratory study. In. In: Fourth International Conference on ACCT 2014, pp. 271–276. IEEE (2014)
13. Rahm, E., Do, H.H.: Data cleaning: problems and current approaches. IEEE Data Eng. Bull. **23**, 3–13 (2000)
14. Dallachiesa, M., Ebaid, A., Eldawy, A., Elmagarmid, A., Ilyas, I.F., Ouzzani, M., Tang, N.: NADEEF: a commodity data cleaning system. In: Proceedings of the 2013 ACM SIGMOD, pp. 541–552. ACM (2013)
15. Fan, W., Geerts, F.: Foundations of data quality management. Synth. Lect. Data Manage. **4**, 1–217 (2012)
16. Dasu, T., Johnson, T.: Exploratory data mining and data cleaning: an overview. In: Exploratory Data Mining and Data Cleaning, pp. 1–16 (2003)
17. Hellerstein, J.M.: Quantitative data cleaning for large databases. United Nations Economic Commission for Europe (UNECE) (2008)
18. Liu, H., Kumar, T.A., Thomas, J.P.: Cleaning framework for big data-object identification and linkage. In: 2015 IEEE International Congress on Big Data, pp. 215–221. IEEE (2015)
19. Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: Proceedings of the Ninth ACM SIGKDD, pp. 39–48. ACM (2003)
20. Wang, J., Kraska, T., Franklin, M.J., Feng, J.: CrowdER: crowdsourcing entity resolution. Proc. VLDB Endowment **5**, 1483–1494 (2012)
21. Müller, H., Freytag, J.C.: Problems, methods, and challenges in comprehensive data cleansing. Professoren des Inst. für Informatik (2005)
22. Krishnan, S., Haas, D., Franklin, M.J., Wu, E.: Towards reliable interactive data cleaning: a user survey and recommendations. In: HILDA@ SIGMOD, p. 9 (2016)
23. Fan, W., Li, J., Ma, S., Tang, N., Yu, W.: Towards certain fixes with editing rules and master data. Proc. VLDB Endowment **3**, 173–184 (2010)
24. Khayyat, Z., Ilyas, I.F., Jindal, A., Madden, S., Ouzzani, M., Papotti, P., Quiané-Ruiz, J.A., Tang, N., Yin, S.: BigDansing: a system for big data cleansing. In: Proceedings of the 2015 ACM SIGMOD, pp. 1215–1230. ACM (2015)
25. Volkovs, M., Chiang, F., Szlichta, J., Miller, R.J.: Continuous data cleaning. In: 2014 IEEE 30th ICDE 2014, pp. 244–255. IEEE (2014)

26. Dai, W., Wardlaw, I., Cui, Y., Mehdi, K., Li, Y., Long, J.: Data profiling technology of data governance regarding big data: review and rethinking. Information Technology: New Generations. AISC, vol. 448, pp. 439–450. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-32467-8_39
27. Gill, R., Singh, J.: A review of contemporary data quality issues in data warehouse ETL environment. J. Today's Ideas Tomorrow's Technol. **2**(2), 153–160 (2014)
28. Gottesheim, W., Mitsch, S., Retschitzegger, W., Schwinger, W., Baumgartner, N.: SemGen—towards a semantic data generator for benchmarking duplicate detectors. In: Xu, J., Yu, G., Zhou, S., Unland, R. (eds.) DASFAA 2011. LNCS, vol. 6637, pp. 490–501. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20244-5_47
29. Papadakis, G., Alexiou, G., Papastefanatos, G., Koutrika, G.: Schema-agnostic vs schema-based configurations for blocking methods on homogeneous data. Proc. VLDB Endowment **9**, 312–323 (2015)
30. Christen, P.: A survey of indexing techniques for scalable record linkage and deduplication. IEEE Trans. Knowl. Data Eng. **24**, 1537–1555 (2012)
31. Bilenko, M., Kamath, B., Mooney, R.J.: Adaptive blocking: Learning to scale up record linkage. In: Sixth International Conference on Data Mining, ICDM 2006, pp. 87–96. IEEE (2006)
32. Michelson, M., Knoblock, C.A.: Learning blocking schemes for record linkage. In: AAAI, pp. 440–445 (2006)

Narrative Annotation of Content for Cultural Legacy Preservation

Pierrick Bruneau(✉)

LIST, 5 Avenue des Hauts-Fourneaux, 4362 Esch-sur-Alzette, Luxembourg
pierrick.bruneau@list.lu

Abstract. An important, yet underestimated, aspect of cultural heritage preservation is the analysis of personal narratives told by citizens. In this paper, we present a server architecture that facilitates multimedia content storage and sharing, along with the management of associated narrative information. Via the exposition of a RESTful interface, the proposed solution enables the collection of textual narratives in raw form, as well as the extraction of related narrative knowledge. We apply it to a corpus related to the time of the European construction in Luxembourg. We disclose details about our conceptual model and implementation, as well as experiments supporting the interest of our approach.

1 Introduction

Collecting and collating personal views and stories of citizens is important in view of preserving the cultural heritage of a time and place. While the wide availability of social media will facilitate this work for future generations when they analyze our times, such means are not available for e.g. the European construction period (roughly 1945–1975). Adapted tools are needed to collect such testimonies by elderly people, as well as facilitate their collation and dissemination.

In this paper we cast our attention towards the time of the European construction in Luxembourg and the surrounding region. This work has been conducted in the context of a funded project in collaboration with elderly people organizations. In this context, witnesses of the time frame of interest (aged between 75 and 85 years old) have been interviewed in French, and their testimonies have been transcribed. An additional French corpus extracted and transformed using online platforms has also been created and studied - more details about corpora may be found in Sect. 4.

Eventually, the collected data is merely more than a set of short texts. To make this narrative data actionable, e.g. allow effective indexing, browsing, or exploitation by web applications, knowledge extraction techniques are needed in order to build relevant metadata from these stories, such as people, places and time frames involved. Hence in this paper we focus on the means to store, process and access such narrative information. More precisely, a dedicated data model and a back-end server are needed in order to model and store the collected stories.

J. Filipe et al. (Eds.): DATA 2017, CCIS 814, pp. 111–127, 2018.
https://doi.org/10.1007/978-3-319-94809-6_6

The rest of this article is organized as follows. Firstly, related work about narrative structures and knowledge is discussed in Sect. 2. Then, in Sect. 3, we define a data model appropriate for storing the collected content, jointly to its narrative metadata extracted by manual and automatic means. We disclose an API that facilitates the manipulation of this model. Next, Sect. 4 describes how knowledge extraction means can use the proposed API in order to annotate the collected content. The collected data is introduced, and the interest of our approach is supported by annotation examples.

2 Related Work

The study of the structure of narratives and stories has been applied to a variety of domains, e.g. emergency response [1], situational awareness [2], or collections of historical documents [3]. A major concern in this domain is to bridge the gap between raw text (i.e. the form under which testimonial stories are generally acquired) and structured information with semantic value, that would enable story linking and advanced queries.

Associating properties to entities and composing them is the core concern of ontology engineering. Well known ontologies include YAGO [4], the Google Knowledge Graph [5], and DBpedia [6]. These ontologies are often used in conjunction with controlled vocabularies such as Dublin Core [7] or FOAF [8], that facilitate the interoperability of data sources, most notably via the RDF representation language.

Rather than an ensemble of unrelated facts, narration implies relationships between atomic facts or events. [1] define a taxonomy of such links (compositionality, causality, correlation and documentation). These links are relevant to our context, but they consider events at a coarse level. Similarly to [3,9], they are mostly concerned by interoperability between different ontologies. Likewise, [2] emphasize link types, with little consideration of specific data structures for events. With close resemblance to the CIDOC CRM [3,10] define explicitly roles (also known as facets in [11]) applying to events (e.g. actors, dates and locations), which are appropriate for historical events in a broad sense (e.g. the French Revolution in [3]), but not for events as constituents of a subjective narrative. The objective is then to propose a standard metadata description space for historical artifacts, rather than exploring the structure of narration.

The contributions by [12] are the most closely related to our work. In their *Narrative Knowledge Representation Language* (NKRL), they define data structures and controlled vocabularies of predicates and links to support the analysis of non-fictional and factual narratives. They avoid the use of the term *story* as it has led to ambiguities in the literature. Rather, They define a set of events and facts as the *fabula*. The *plot* level adds chronological, logical and coherence links between events. The *presentation* level is about the form in which plots are shown. Some related work in narrative analysis and storytelling is concerned with mapping arbitrary stories to a classical narrative structure [13,14]. In our work, stories are potentially made of anecdotal testimonies, and as such cannot be expected to match these structures. More abstract properties, such as sentiment attached to stories, were also extracted in [15] in order to analyze the structure of books.

In [16], a simplified version of the model by [12], and premises of its implementation are presented. However, in the latter the narrative knowledge is laid out in the same schema as entities describing pieces of content. This complicates the extraction of narrative knowledge by multiple agents, e.g. algorithms or human annotators. In the multimedia processing literature, when establishing benchmarks for evaluation, it is common to decorrelate the managed content from its annotations [17]. In the context of the present paper, narrative knowledge could thus be considered as a special kind of annotations.

The way narrative entities, relationships and predicates are extracted is seldom considered in the literature reported above. Some authors explicitly assume that this mapping has to be performed manually [11], or via crowdsourcing [18]. Wikipedia page structure has also been exploited in [4]. Alternatively, a term-based heuristic is used in [19] to determine links between events, and the use of *Natural Language Processing* (NLP) techniques such as *Named Entity Recognition* (NER) to automatically extract facts and events has been evaluated in [3,20]. Entity types in event models such as SEM [2] are closely related to types extracted by standard NER methods such as [21] (e.g. people, locations, dates).

3 Narrative Annotation System Description

The model described in this section answers to complementary concerns: the formalization of narrative knowledge, and the management of multimedia content. The latter is a crucial aspect, as the content is the core set of legacy objects that are to be preserved (e.g. textual or audio testimonies, photographs).

On the other hand, we aim at facilitating high-level semantic tasks over the managed content. For example, users should be able to find content that relates to a given place or person, as well as find redundant or contradictory events. An adapted narrative knowledge model is thus necessary.

With regard to these context and constraints, we define a two-level model:

- The first level focuses on narrative modelling, with the definition of real-world entities such as places, people or times, as well as possible relationships among these structures. It elaborates on previous work on the topic [12,16]. We present it in Sect. 3.1.
- The second level is focused on content description. Aside wrappers to the managed pieces of content (e.g. stories, audio, images), it defines the *annotation* data structure, that is the core articulation between the abstract semantic model and the managed content. This facet of our system is disclosed in Sect. 3.2.

Section 3.3 focuses on the API that enables access to these models by tier applications.

3.1 Narrative Description Model

The role of the narrative data model exposed in this section is to describe entities such as places and people, and relationships that exist between these. Figure 1 shows the proposed model. We consider that instances from this model exist *per se*, i.e. independently of a specific piece of content.

The data model in Fig. 1 is heavily inspired by the model underlying NKRL [12], but exhibits decisive distinctions. The proposed structure was designed with flexibility in mind. For example, it enables partial specification - a typical narrative may occasionally omit spatial and/or temporal specifications. Notably, the proposed *TimeSpan* format allows loose specification, with all fields except year being optional. Both points and intervals in time can be described with the same format, simply by equaling *From* and *To* respective fields.

The layout of the schema in Fig. 1 emphasizes the complexity of narrative data: both simple (e.g. a person pictured in a multimedia document) and complex (e.g. plot structure in a story text) entities are supported.

In Fig. 1, we note that fields can directly hold references to other entities. This feature is considered in abstract terms for now. The means for its implementation are discussed at length in Sect. 4. When a field may feature several entity types, the generic *Entity* is indicated. It may be thought as the root *Object* type in some object-oriented programming languages, as all entity types in Fig. 1 derive directly from it.

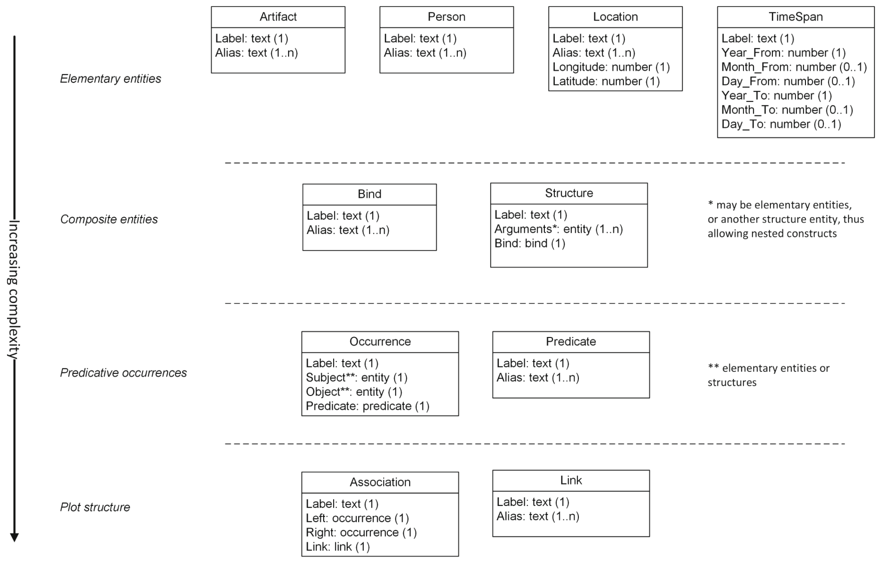


Fig. 1. Narrative data model. From top to bottom, entities are ordered by increasing complexity.

Entities from types depicted in Fig. 1 exist as single instances in the database. The *Alias* fields enable a single underlying instance to be designated by multiple writings. It is then up to the user managing the database (or knowledge extraction algorithms) to establish which alternate writings refer to which identical underlying instance.

This model was designed with expressivity and flexibility in mind, rather than mere consistency. This approach distinguishes from many works in ontology engineering which are mainly focused towards reasoning, i.e. inference of novel facts that can be deduced from the current fact base. The latter view can be valuable when actionable data is sought [1,2]. In narration, the focus is not so much on deduction than on facilitating the access and the presentation of the data. In our approach, ensuring the stored narrative knowledge is consistent accross the database is delegated to users and tier applications.

Following our simplified schema, complex narrative structures are stored using nested structures. Figure 2 shows an instanciation example. The complexity mentioned in Fig. 1 can be seen as the height that instances of a given entity generally takes.

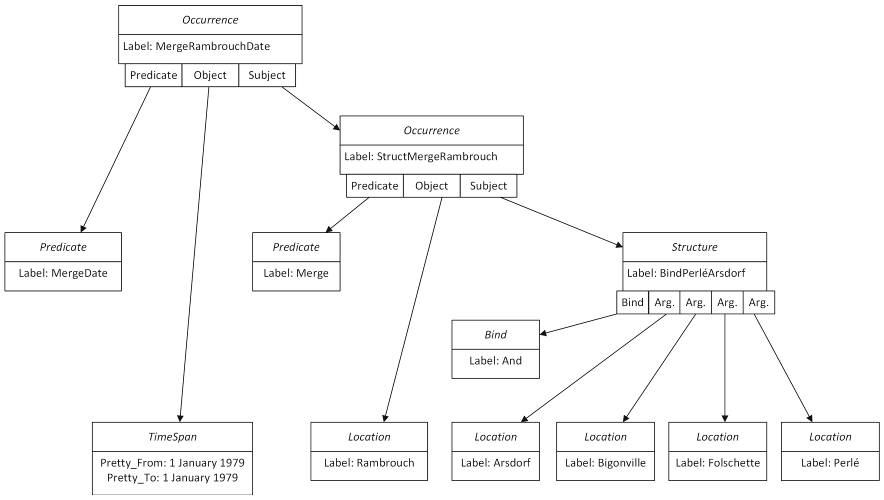


Fig. 2. Narrative instances extracted from the French text: *Perlé fut une commune jusqu'au 1er janvier 1979, date à laquelle elle a fusionné avec les communes d'Arsdorf, Bigonville et Folschette pour former la nouvelle commune de Rambrouch*. Entity reference fields are represented as sockets at the bottom of instances, with arrows pointing to the linked instance.

Our way of encoding predicative occurrences is reminiscent of RDF-based ontologies, that rely on binary relations, and reification to represent more complex semantics. In the context of general ontologies, [22] indeed show that using reification, complex facts can be unfolded as several atomic facts. In brief, with reification, an instance of a binary relation $a\mathcal{R}_1b$ can act as the argument of another relation, effectively allowing e.g. $(a\mathcal{R}_1 b)\mathcal{R}_2 c$. This design has been subject to debate in the literature. For example, [12] advocates the usage of pure n-ary relations. Our eventual choice has been motivated by its greater flexibility, and better compliance with the technologies chosen for its implementation (see Sect. 4).

An important terminological nuance lies between our schema proposition and those also using reified facts: in the latter, properties are linked to entities by a static set of *predicates*, when in our proposition the *predicate* is a full-fledged entity type, with no constraints on the values its instances take. This choice is guided by the need to enable greater expressivity, as required in the context of narratives. Dealing with such an open vocabulary yields additional difficulties, addressed using NLP means as discussed to a greater extent in Sect. 4.

3.2 Content Description Model

In [16], the narrative entities introduced in Sect. 3.1 are laid out in the same schema as entities describing pieces of content. Here we choose to separate entities that support the narrative knowledge formalization, from those that describe the actual managed content. The content schema is given in Fig. 3.

We distinguish two main types of documents: stories, that are essentially made of a string of text, and media, that wrap actual media files along with a

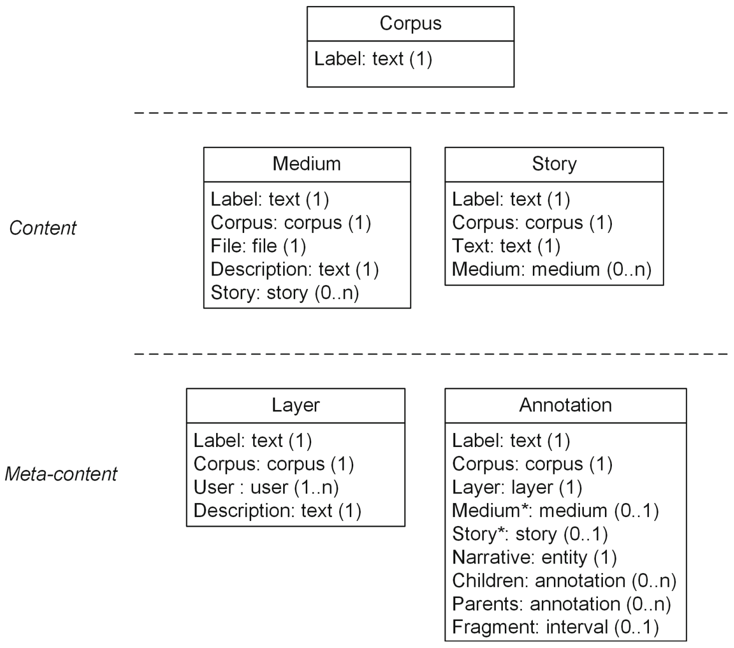


Fig. 3. Content description model. It distinguishes the actual managed content, from meta-content, essentially made from annotations.

formal description. Optional direct links may exist between these entities: one or more stories may act as the caption for a medium, as well as one or more media may decorate a given story.

Following principles developed in the Camomile project [17], the articulation between narrative entities discussed in Sect. 3.1 and content description is made by *annotations*. Roughly, via their *narrative* field, they associate narrative entities from the schema in Fig. 1 to whole or fragments of managed multimedia content. Specifically, the *fragment* field, basically an array of integers, optionally allows the annotation to be specific of a fragment of a document. Its meaning depends on the context - e.g. starting and ending character positions for a story, beginning and ending timestamp in seconds for audio and video, or a pixel bounding box for pictures. A simple example illustrating the distinction between content, annotations and narrative entities in given in Fig. 4.

The root concept in Fig. 3, the *corpus*, abstracts a group of homogeneous documents. This will enable the separate treatment of the two distinct corpora presented in Sect. 4.1. *Layer*s are meant to gather annotations of identical nature, w.r.t. content of a given corpus. For example annotations by a given user or group of users during a specific manual annotation session, or by a given automatic annotation algorithm, may fall in distinct layers. This addresses a very pragmatic concern: the fact that the same story or medium might be annotated several times, possibly differently, by multiple agents. A layer may assemble annotations of several narrative types, covering any part of the corpus (i.e. its media and stories).

In addition to the guidelines introduced in [17], the annotation entity in Fig. 3 offers the possibility to form annotation hierarchies, via the *children* and *parents* fields. The intent is to mirror the potential complexity of constructions following the narrative data model (see e.g. Fig. 2), and facilitate retrieval operations in the context of annotation tasks.

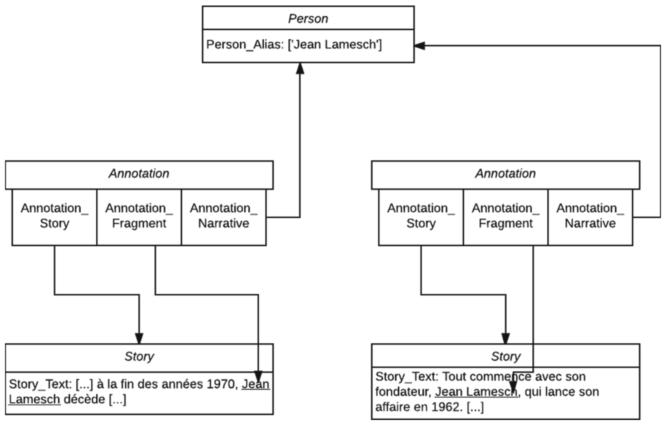

Person
Person_Alias: ['Jean Lamesch']
Annotation
Annotation_ Story
Annotation_ Fragment
Annotation_ Narrative
Annotation
Annotation_ Story
Annotation_ Fragment
Annotation_ Narrative
Story
Story_Text: [...] à la fin des années 1970, Jean Lamesch décède [...]
Story
Story_Text: Tout commence avec son fondateur, Jean Lamesch, qui lance son affaire en 1962. [...]

Fig. 4. Example showing the link between two stories (on bottom, extracted from the *web* corpus described in Sect. 4.1) and the same underlying entity (person, on top) ensured by annotations.

3.3 API Description

The models exposed in Figs. 1 and 3 are implemented under the Drupal Content Management System [16]. Specifically, entity instances are implemented as *nodes* in Drupal. Our narrative and content entities are implemented as specializations of this baseline, notably supporting optional and variable-sized fields. Beyond primitive types (e.g. text strings, numbers), Drupal offers the possibility to define fields holding *entity references*: this feature enables links between entities as displayed in Fig. 2 for example.

Drupal also features a RESTful module [23], that allows to bind entity types as defined in a Drupal backend to REST API routes. In brief, REST is a protocol based on the HTTP protocol and the JSON format, to allow consumption, creation and update of the data held by a Drupal backend. The RESTful module manages the conversion to and from entity references and node IDs. The general architecture in which the REST API and the underlying model situate is described by Fig. 5. In brief, the REST API is the interface to the backend data stored according to the schema described in Sects. 3.1 and 3.2. It enables the consumption and production of the data by both automatic (e.g. NER techniques) and interactive programs (e.g. web-based annotation tools).

Figures 6 and 7 show the REST API routes mirroring the entities shown in Figs. 1 and 3, respectively. Without any filter, a route returns all entities of the associated type. Query filters can be applied following the syntax presented in [24], using node IDs or primitive values. The *annotation_narrative_type* filter in the *annotation* route allows filtering according to entity types shown in Fig. 1. *annotation_narrative_type* and *annotation_narrative* filters are matched by inspecting the entity referenced by the respective *narrative* field.

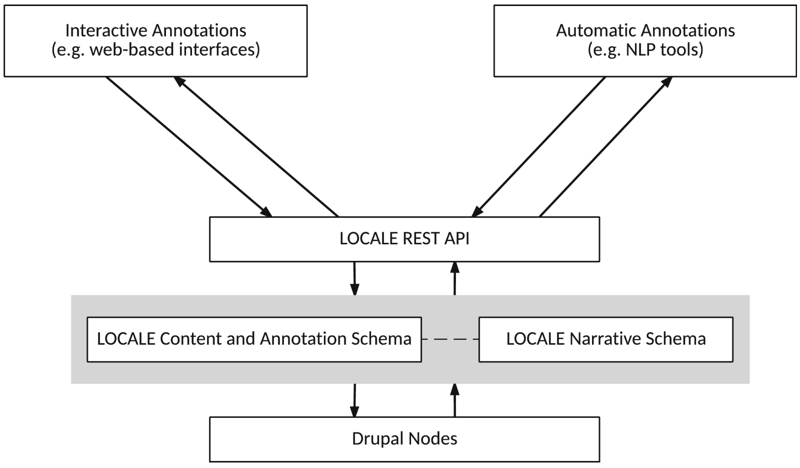


Fig. 5. General architecture of the LOCALE platform, from the REST API perspective.

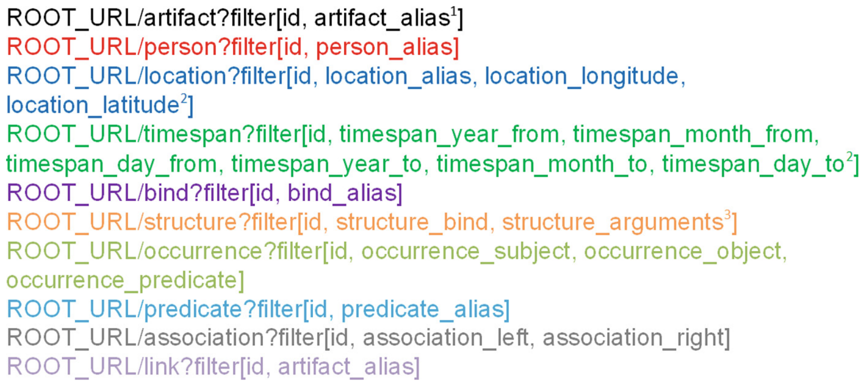

ROOT_URL/artifact?filter[id, artifact_alias^1]
ROOT_URL/person?filter[id, person_alias]
ROOT_URL/location?filter[id, location_alias, location_longitude, location_latitude^2]
ROOT_URL/timespan?filter[id, timespan_year_from, timespan_month_from, timespan_day_from, timespan_year_to, timespan_month_to, timespan_day_to^2]
ROOT_URL/bind?filter[id, bind_alias]
ROOT_URL/structure?filter[id, structure_bind, structure_arguments^3]
ROOT_URL/occurrence?filter[id, occurrence_subject, occurrence_object, occurrence_predicate]
ROOT_URL/predicate?filter[id, predicate_alias]
ROOT_URL/association?filter[id, association_left, association_right]
ROOT_URL/link?filter[id, artifact_alias]

Fig. 6. Routes related to narrative information exposed by the REST API. Superscripted numbers are explained in the text of Sect. 3.3.

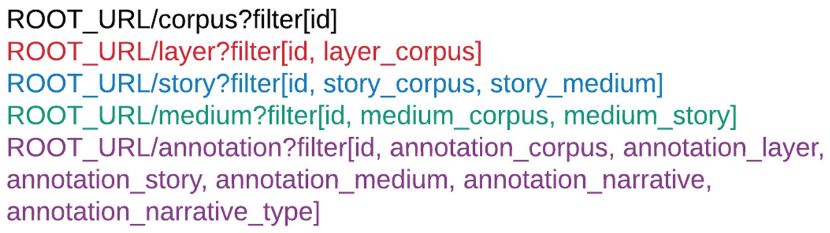
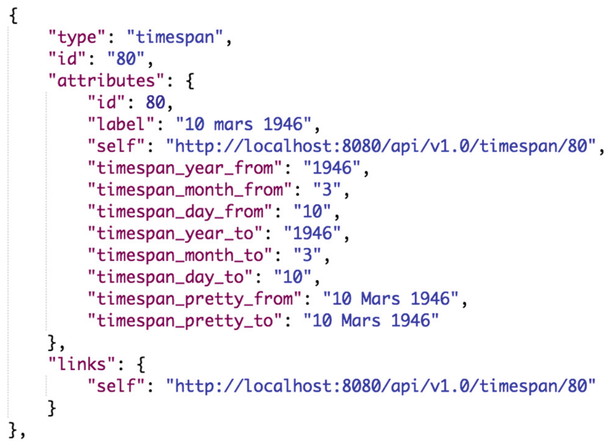

ROOT_URL/corpus?filter[id]
ROOT_URL/layer?filter[id, layer_corpus]
ROOT_URL/story?filter[id, story_corpus, story_medium]
ROOT_URL/medium?filter[id, medium_corpus, medium_story]
ROOT_URL/annotation?filter[id, annotation_corpus, annotation_layer, annotation_story, annotation_medium, annotation_narrative, annotation_narrative_type]

Fig. 7. Routes related to content exposed by the REST API.

The routes described in Fig. 7 return information describing managed media and their annotations. Actual narrative entities behind annotations, i.e. semantic values that annotate documents, can be accessed via the routes shown in Fig. 6 using the fetched IDs. The *Location* entities can be queried by filling the *Longitude* and *Latitude* filters. *Longitude* and *Latitude* can be specified as relative floats. By convention, positive latitudes (resp. longitudes) denote the northern part of the Earth (resp. eastern part w.r.t. Greenwich meridian). Filters mirroring the *TimeSpan* entity fields can also be specified.

All routes in Figs. 6 and 7 support the GET (i.e. retrieving entities) verb, as well as PATCH (i.e. updating an entity) or POST (i.e. creating a new entity) whenever appropriate. The Drupal RESTful modules allows the fairly straightforward customization of the input and outputs of the routes defined. We used this facility to implement a *pretty* date format, that allows the conversion of textual dates to and from the schema defined in Fig. 1 (see Fig. 8 for an example).



```
{
    "type": "timespan",
    "id": "80",
    "attributes": {
        "id": 80,
        "label": "10 mars 1946",
        "self": "http://localhost:8080/api/v1.0/timespan/80",
        "timespan_year_from": "1946",
        "timespan_month_from": "3",
        "timespan_day_from": "10",
        "timespan_year_to": "1946",
        "timespan_month_to": "3",
        "timespan_day_to": "10",
        "timespan_pretty_from": "10 Mars 1946",
        "timespan_pretty_to": "10 Mars 1946"
    },
    "links": {
        "self": "http://localhost:8080/api/v1.0/timespan/80"
    }
},
```

Fig. 8. Date record example, as retrieved from the *ROOT_URL/timespan* route. The POST and PATCH verbs also parse dates provided under the *pretty* format, if provided.

In the following list, we elaborate about specific points highlighted in Fig. 6:

1. Alias fields have variable size: as a result, e.g. *filter[artifact_alias]* = *STRING_VALUE* matches against any value of the respective array.
2. Filtering other than equality can be specified e.g. with *filter[location_{longitude|latitude}][operator]=FLOAT_RADIUS_VALUE*. This also applies to point 1, where an integer from 0 to n could indicate approximate matching using edit distance.
3. Arguments may be queried directly via their node ID.

4 Experiments

4.1 Corpora Description

First we describe two corpora that have been loaded in our backend:

– 266 short stories (avg. 140 characters) related to the period of 1945–1975 in Luxembourg and the surrounding region. The stories were selected and extracted automatically using the the Google Custom Search Engine API[1]. More precisely, the Google API was invoked for a list of heuristical queries about well known Luxemburgish locations or companies (e.g. *Kirchberg, Luxair*) for the targeted time frame. Results originate from several web portals (i.e. Wikipedia, http://www.industrie.lu, http://www.vdl.lu). Each story is associated to a date and a location name. This kind of indexing is easily handled by the model described in Fig. 1. Each location name has been mapped with latitude and longitude using the Google Maps Geocoding API (See footnote 1). We refer to this corpus as *web* later on. This set of stories is described to further extent in [25].

[1] https://developers.google.com.

– Interviews were conducted with elderly people that have lived in Luxembourg in the time frame of interest (1945–1975). We used photograph collections from the spatio-temporal time frame of interest in order to trigger memories, as well as the *web* subset. We obtained approximately 5 h of audio recordings, by 5 participants, among which 2 h have been manually transcribed and segmented into 9 stories (avg. 3964 characters). We refer to this subset as *interviews* later on.

Unique labels are simple to initialize for *Location* and *TimeSpan* narrative entities (the name itself and the date format illustrated in Fig. 8, respectively). As distinct stories might have the same *n* initial words, in our implementation, unique story labels are generated as MD5 hashes [26] from the respective text.

4.2 Knowledge Extraction

As exposed in Sect. 2, the most frequent setting in the literature is to consider that the mapping of a presentation to a plot structure is performed manually. In this section, we describe means to extract, at least approximately, the narrative information out of this initial representation. Performing this automatic mapping operation can have various utilities in the context of the project described in the introduction. First, as mentioned in Sect. 4.1, testimonies in our data corpus are recorded and transcribed manually, but exhibit no structure that facilitate their presentation in context, and exploration. Offline extraction of the narrative structure would avoid tedious manual efforts. As mentioned in the introduction, our research context copes with testimonies collected in French. This constrained the technologies discussed later on in this section.

Named Entity Recognition consists in detecting entities such as people and places automatically in text. For example, the LIA tools [21] recognize people, locations, organizations, socio-political groups, quantities, dates and products in French text. Such facilities can then be a crucial initial step towards feeding the model described in Fig. 1, of which people, places and time specifications are the core. The most renowned general purpose natural language processing system, the Stanford CoreNLP suite [27], also provides such NER functionalities for several languages including French.

In order to structure recognized entities and annotations according to the schema described in Fig. 3, and possibly extract non-named entities (i.e. *Artifacts* in Fig. 1), syntactic cues are needed. *Part-Of-Speech* (POS) tagging is about estimating the function of words in text (e.g. adjective, verb, determinant). *Semantic Role Labeling* (SRL) builds upon POS-tagging in order to extract higher-order structures (e.g. subject, object, verbal forms), that are very close to the syntactic cues expected in our model. Actually this is not surprising insofar as the same seminal references in language analysis are foundational both for narratology [12] and SRL [28]. POS-tagging facilities are available in French both in the LIA tools [21] and the CoreNLP suite [27]. The latter also offers facilities in SRL, which are used by examples shown in Sect. 4.

NLP tools presented above allow extracting predicates and structural information. However, as we consider open vocabularies for *Predicate*, *Bind* and *Link* instances, semantics still have to be attached to them, e.g. allowing queries for instances similar to a template. Prepositions and coordination markers take their values in sufficiently small sets, so term-based queries are acceptable then. But as verbs are very numerous, a proxy is needed to allow queries for similar *Predicate* instances. Word embedding techniques associates a value in a high-dimensional numeric space to words, in a way that reflects word semantics [29]. We propose to use the implicit structure reflected by the word embedding space. Such mapping functions can be implemented locally using models from libraries such as TensorFlow [30] trained with a corpus in French. We used the complete French Wikipedia archive[2]. It contains approximately 2.3M articles in XML format. We converted them to a plain text format as expected by the training algorithm using the tool proposed by [31]. Punctuation and other non-textual characters were then removed using regular expressions. No stemming is required as all forms of a given word are embedded separately, and generally end up in close vicinity to each other.

4.3 Automatic Annotation Scenario

In this section, we illustrate how our REST API can be used in conjunction with NLP means in order to extract and attach narrative knowledge to the managed data. The following excerpt from the *interviews* corpus is used to support the illustration: *Quand le théâtre national a été construit, un grand architecte Français est intervenu. Il a aussi travaillé au Châtelet à Paris. Il voulait faire un grand balcon, comme au Châtelet. Alors un Luxembourgeois a dit "oh non, ça ne va pas, il faut construire une loge pour la Grande Duchesse". Alors, au milieu, une loge a été construite, mais les Grands Ducs n'y vont jamais. Ils préfèrent être au premier rang, là où on voit le mieux. Je me souviens de l'ouverture du théâtre, on y était, il y avait le fameux danseur mondialement connu, Maurice Béjart. Il a fait le ballet de Ravel. Il est venu pour l'inauguration officielle du théâtre.* In brief, the story is about the new theatre that opened in Luxembourg in 1964.

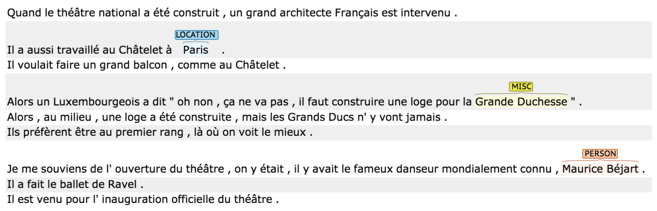

Quand le théâtre national a été construit , un grand architecte Français est intervenu .
Il a aussi travaillé au Châtelet à Paris .
Il voulait faire un grand balcon , comme au Châtelet .
Alors un Luxembourgeois a dit " oh non , ça ne va pas , il faut construire une loge pour la Grande Duchesse " .
Alors , au milieu , une loge a été construite , mais les Grands Ducs n' y vont jamais .
Ils préfèrent être au premier rang , là où on voit le mieux .
Je me souviens de l' ouverture du théâtre , on y était , il y avait le fameux danseur mondialement connu , Maurice Béjart .
Il a fait le ballet de Ravel .
Il est venu pour l' inauguration officielle du théâtre .

Fig. 9. Named entities extracted by the CoreNLP server, visualized using *brat* [32].

[2] https://dumps.wikimedia.org/frwiki/latest/.

Using a script, this piece of content was retrieved using the *story* REST route (see Fig. 7), and submitted to a local CoreNLP server [27] in order to tentatively identify named entities. Extracted entities are shown in Fig. 9. The extracted set is neither complete (e.g. *Grands Ducs* and *théâtre national* are not detected as *Person* and *Location* entities, respectively), nor fully accurate (e.g. *Grande Duchesse* is set in the *miscellaneous* category, when it is truly a *Person* entity). Nevertheless, the extracted entities can be used to create respective *Person* and *Location* entities. The resulting narrative and content entity structure, mirroring the example in Fig. 4, is shown in Fig. 10, and stored using our REST API.

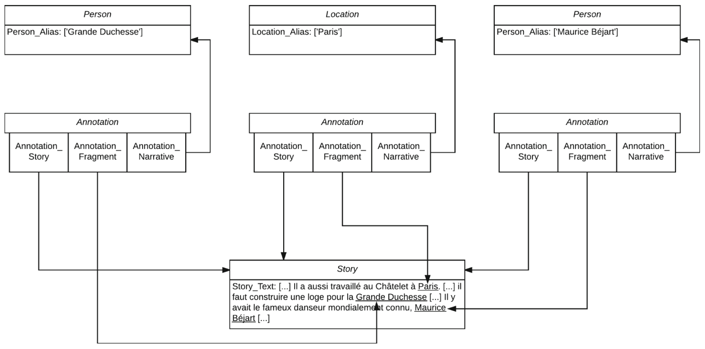

Person
Person_Alias: ['Grande Duchesse']
Location
Location_Alias: ['Paris']
Person
Person_Alias: ['Maurice Béjart']
Annotation
Annotation_ Story
Annotation_ Fragment
Annotation_ Narrative
Story
Story_Text: [...] Il a aussi travaillé au Châtelet à Paris. [...] il faut construire une loge pour la Grande Duchesse [...] Il y avait le fameux danseur mondialement connu, Maurice Béjart [...]

Fig. 10. Annotations and narrative entities resulting from the NER.

In view of extracting predicative occurences that involve the named entities identified above, we apply the SRL facilities available in the CoreNLP server to the respective sentences. The result from this analysis is depicted in Fig. 11. Then we apply the following procedure:

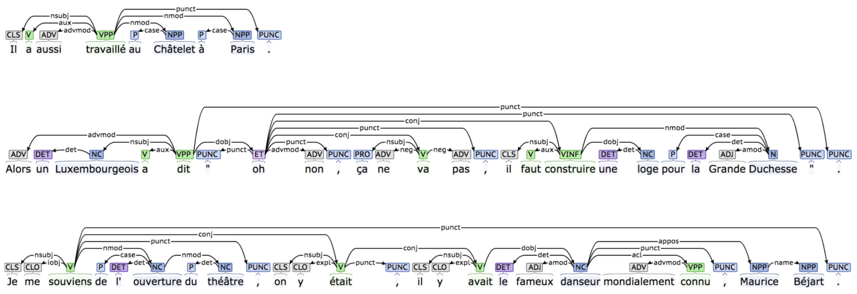

Il a aussi travaillé au Châtelet à Paris .
Alors un Luxembourgeois a dit " oh non , ça ne va pas , il faut construire une loge pour la Grande Duchesse " .
Je me souviens de l' ouverture du théâtre , on y était , il y avait le fameux danseur mondialement connu , Maurice Béjart .
nsubj
aux
advmod
punct
nmod
case
det
dobj
conj
neg
expl
iobj
amod
acl
appos
name

Fig. 11. SRL results, visualized using *brat* [32].

- Identify the verbal tokens that are parent of at least a group where a noun is involved (e.g. *travaillé*, *dit*, *construire*, *souviens* and *avait* in Fig. 11).
- Form predicative occurences with predicates that have subject (*nsubj* in Fig. 11) and/or object (*dobj* in Fig. 11) links in the SRL output.

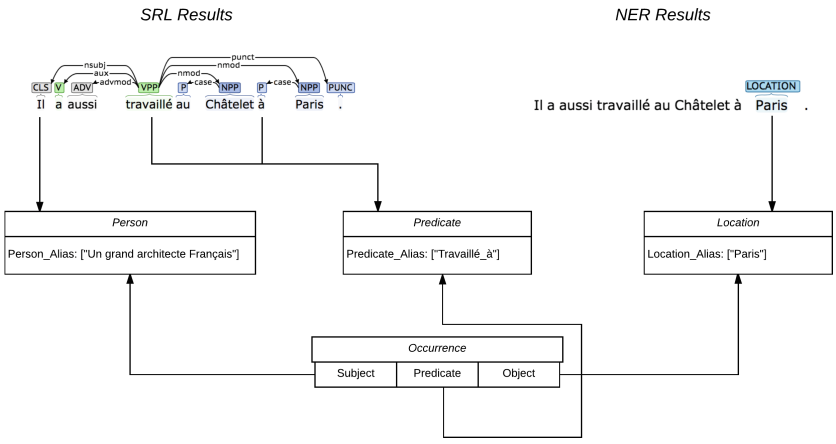
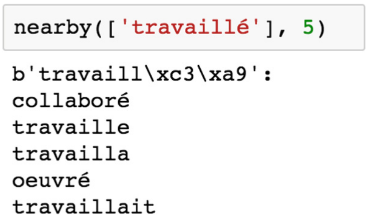

SRL Results
NER Results
nsubj
aux
advmod
punct
nmod
nmod
case
case
CLS
V
ADV
VPP
P
NPP
P
NPP
PUNC
Il a aussi travaillé au Châtelet à Paris .
LOCATION
Il a aussi travaillé au Châtelet à Paris .
Person
Person_Alias: ["Un grand architecte Français"]
Predicate
Predicate_Alias: ["Travaillé_à"]
Location
Location_Alias: ["Paris"]
Occurrence
Subject
Predicate
Object

Fig. 12. Construction of a predicative occurrence using SRL and NER results.

– If nominal modifiers are found (*nmod* in Fig. 11), create predicates that concatenate the verbal token and the respective preposition - if they do not already exist - and create predicate occurences accordingly.
– If coordination links are found between predicates (i.e. *conj* in Fig. 11), create associations between occurences accordingly.

Figure 12 illustrates the construction of predicative occurences using this procedure. Annotations linking the occurrences to the respective content are then created accordingly. As discussed in Sect. 4.2, prepositions (e.g. *à*, *pour*, *du*) and coordination markers (e.g. comma, *mais*) take their values in small sets. When appropriate, we concatenate the latter with the respective predicates. Later on, when querying the backend for similar predicative occurences, we may deparse the verb from the constructed predicate, and perform a query using nearest neighbors in a word embedding space as index. Figure 13 gives an example of such a query.



```
nearby(['travaillé'], 5)
```

```
b'travaill\xc3\xa9':
collaboré
travaille
travailla
oeuvré
travaillait
```

Fig. 13. The 5 nearest neighbors of the word *travaillé* (passive form of the verb *work*) in the word embedding space.

5 Conclusion

We described a schema and an API that facilitate the storage, processing and sharing of personal narratives related to the period of 1945–1975 in Luxembourg. We tested it with real data collected from the web and interviews conducted with witnesses of the spatio-temporal time frame of interest, and showed how our API can be used in conjunction to NLP means in order to extract and manage narrative knowledge. As NLP tools are prone to errors and omissions, semi-automatic means to annotate and curate the corpora presented in Sect. 4.1 would be the most natural extension to this work. A variant of the latter application would be the semi-automatic input of a story. When a user types a story in an interface, text would be sent on the fly to processing services. Based upon extracted entities, related stories can then be displayed live to the client as a contextual help.

A key extension would be also to enable the resolution of conflicts between stories and plots, as well as the resolution of entities having alternative writings. Classical use of reasoning is to enable deduction of novel facts if adequate rules are defined [4], but this range of techniques has also been used to detect contradictions [33].

Finally, we will study the issue of subjectivity: personal narratives often contain implicit references to the narrator (e.g. me, my brother). Experiments in Sect. 4.3 show these references can only be indirectly detected. Simple heuristics could be implemented by using a closed list of keywords along with the extracted cues.

Acknowledgements. This work was achieved in the context of the LOCALE project, funded by the Luxembourgish *Fonds National de la Recherche*. It used data from the *Centre National de l'Audiovisuel*. We would also like to thank elderly volunteers from *AMIPERAS* and *Konviksgaart* organizations, who contributed to the creation of the *interviews* corpus.

References

1. Scherp, A., Franz, T., Saathoff, C., Staab, S.: F-A model of events based on the foundational ontology DOLCE+DnSUltralite. In: K-CAP 2009, pp. 137–144 (2009)
2. Van Hage, W., et al.: Abstracting and reasoning over ship trajectories and web data with the Simple Event Model (SEM). Multimedia Tools Appl. **57**(1), 175–197 (2012)
3. Segers, R., et al.: Hacking history: automatic historical event extraction for enriching cultural heritage multimedia collections. In: K-CAP 2011 (2011)
4. Suchanek, F.M., et al.: Yago: a large ontology from Wikipedia and WordNet. Web Semant. Sci. Serv. Agents WWW **6**(3), 203–217 (2008)
5. Google: Introducing the knowledge graph (2012). http://tinyurl.com/zofw8fb
6. Mendes, P.N., Jakob, M., García-Silva, A., Bizer, C.: DBpedia spotlight: shedding light on the web of documents. In: ACM Semantics, pp. 1–8 (2011)
7. Weibel, S., Kunze, J., Lagoze, C., Wolf, M.: Dublin core metadata for resource discovery (no. rfc 2413) (1998)

8. Brickley, D., Miller, L.: Foaf vocabulary specification 0.91. Technical report, ILRT Bristol (2007)
9. van der Meij, L., Isaac, A., Zinn, C.: A web-based repository service for vocabularies and alignments in the cultural heritage domain. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010. LNCS, vol. 6088, pp. 394–409. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13486-9_27
10. Doerr, M.: The CIDOC conceptual reference module: an ontological approach to semantic interoperability of metadata. AI Mag. **24**(3), 75–92 (2003)
11. Mulholland, P., Wolff, A., Collins, T.: Curate and storyspace: an ontology and web-based environment for describing curatorial narratives. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 748–762. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30284-8_57
12. Zarri, G.: Representation and Management of Narrative Information. Theoretical Principles and Implementation. Springer, London (2009). https://doi.org/10.1007/978-1-84800-078-0
13. Tilley, A.: Plot Snakes and the Dynamics of Narrative Experience. University Press of Florida, Gainesville (1992)
14. Yeung, C., et al.: A knowledge extraction and representation system for narrative analysis in the construction industry. Expert Syst. Appl. **41**(13), 5710–5722 (2014)
15. Min, S., Park, J.: Mapping out narrative structures and dynamics using networks and textual information. arXiv preprint arXiv:1604.03029 (2016)
16. Bruneau, P., Parisot, O., Tamisier, T.: Storing and processing personal narratives in the context of cultural legacy preservation. In: DATA (2017)
17. Poignant, J., Budnik, M., Bredin, H., Barras, C., Stefas, M., Bruneau, P., Adda, G., Besacier, L., Ekenel, H., Francopoulo, G., Hernando, J., Mariani, J., Morros, R., Quénot, G., Rosset, S., Tamisier, T.: The CAMOMILE collaborative annotation platform for multi-modal, multi-lingual and multi-media documents. In: LREC Conference (2016)
18. Bollacker, K., et al.: Freebase: a collaboratively created graph database for structuring human knowledge. In: SIGMOD, pp. 1247–1250 (2008)
19. Gaeta, A., Gaeta, M., Guarino, G.: RST-based methodology to enrich the design of digital storytelling. In: IEEE INCOS 2015, pp. 720–725 (2014)
20. Van Hooland, S., et al.: Exploring entity recognition and disambiguation for cultural heritage collections. Digit. Scholarsh. Humanit. **30**(2), 262–279 (2015)
21. Favre, B., Béchet, F., Nocéra, P.: Robust named entity extraction from large spoken archives. In: HLT/EMNLP 2005, pp. 491–498 (2005)
22. Hoffart, J., Suchanek, F.M., Berberich, K., Weikum, G.: YAGO2: a spatially and temporally enhanced knowledge base from Wikipedia. Artif. Intell. **194**, 28–61 (2013)
23. Burstein, A., Bosch, M.: RESTful best practices for Drupal (2014). https://github.com/RESTful-Drupal/restful
24. Burstein, A., Bosch, M.: Applying a query filter (2017). https://github.com/RESTful-Drupal/restful/blob/7.x-2.x/docs/api_url.md#applying-a-query-filter
25. Parisot, O., Tamisier, T.: A corpus of narratives related to Luxembourg for the period 1945–1975. In: 28th International Workshop on Database and Expert Systems Applications, pp .113–117 (2017)
26. Rivest, R.: The MD5 Message-Digest Algorithm. RFC 1321, MIT and RSA Data Security (1992)

27. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J.R., Bethard, S., McClosky, D.: The stanford CoreNLP natural language processing toolkit. In: ACM Semantics, pp. 55–60 (2014)
28. Jurafsky, D., Martin, J.H.: Semantic role labeling. In: Speech and Language Processing. Pearson (2014)
29. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS, pp. 3111–3119 (2013)
30. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: TensorFlow: a system for large-scale machine learning. In: 12th USENIX conference on Operating Systems Design and Implementation, pp. 265–283 (2016)
31. Attardi, G.: A tool for extracting plain text from Wikipedia dumps (2016). https://github.com/attardi/wikiextractor
32. Stenetorp, P., Pyysalo, S., Topić, G., Ohta, T., Ananiadou, S., Tsujii, J.: BRAT: a web-based tool for NLP-assisted text annotation. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics, pp. 102–107 (2012)
33. Paulheim, H.: Knowledge graph refinement: a survey of approaches and evaluation methods. In: Semantic Web, pp. 1–20 (2016)

Determining Appropriate Large Object Stores with a Multi-criteria Approach

Uwe Hohenstein[1(✉)], Spyridon V. Gogouvitis[2(✉)], and Michael C. Jaeger[1(✉)]

[1] Corporate Technology, Siemens AG, Otto-Hahn-Ring 6, 81730 Munich, Germany
{uwe.hohenstein,michael.c.jaeger}@siemens.com

[2] Mobility Division, Siemens AG, Otto-Hahn-Ring 6, 81730 Munich, Germany
gogouvitis@siemens.com

Abstract. The area of storage solutions is becoming more and more heterogeneous. Even in the case of relational databases, there are several offerings, which differ from vendor to vendor and are offered for different deployments like on-premises or in the Cloud, as Platform-as-a-Service (PaaS) or as a special Virtual Machine on the Infrastructure-as-a-Service (IaaS) level. Beyond traditional relational databases, the NoSQL idea has gained a lot of attraction. Indeed, there are various services and products available from several providers. Each storage solution has virtues of its own even within the same product category for certain aspects. For example, some systems are offered as cloud services and pursue a pay-as-you-go principle without upfront investments or license costs. Others can be installed on premises, thus achieving higher privacy and security. Some store redundantly to achieve high reliability for higher costs. This paper suggests a multi-criteria approach for finding appropriate storage for *large* objects. Large objects might be, for instance, images of virtual machines, high resolution analysis images, or consumer videos. Multi-criteria means that individual storage requirements can be attached to objects and containers having the overall goal in mind to relieve applications from the burden to find corresponding appropriate storage systems. For efficient storage and retrieval, a metadata-based approach is presented that relies on an association with storage objects and containers. The heterogeneity of involved systems and their interfaces is handled by a federation approach that allows for transparent usage of several storages in parallel. All together applications benefit from the specific advantages of particular storage solutions for specific problems. In particular, the paper presents the required extensions for an object storage developed by the VISION Cloud project.

Keywords: Cloud storage · Federation · Multi-criteria · NoSQL

1 Introduction

According to the National Institute of Standards and Technology (NIST) [18] Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. One important resource that can be provisioned by a Cloud is obviously storage. In particular, Cloud storage offers all the benefits that are described by [11] for Cloud computing in general. A user can ask for virtually unlimited storage space, provisioned in a couple of minutes, thereby paying only for the provisioned storage. This is known as the pay-as-you-go principle and implies that no long-term commitments and no upfront costs apply to usage. Moreover, users do not have to take care about software licenses (e.g., as for database products) since these are already included in the usage fees.

Cloud storage solutions usually include Blob-stores for large unstructured objects and NoSQL databases [22], the latter trying to establish an antipole to traditional relational (i.e., SQL) databases. In fact, the term "NoSQL" stands more moderately for "not only SQL". NoSQL databases abdicate the well-known transactional ACID properties of SQL databases in favor of highly distributing data across several computer nodes, thus benefiting from parallelism. This perfectly fits to cloud computing environments. Furthermore, NoSQL databases offer more flexibility by not relying on a pre-defined, fixed table schema. Schema-less data can be stored in flexible NoSQL structures, i.e., the kind of data and its structure does not have to be known in advance.

Despite new arising storage technologies, well-known relational databases and providers are also offered by cloud providers. Consequently, relational databases are also a kind of cloud storage in a broader sense.

In particular, NoSQL is again a placeholder for a lot of different approaches and products, also covering "elder" technologies such as object-oriented or XML databases. Consequently, an increasing heterogeneity of storage technologies is currently arising, also keeping in mind the product or vendor specific and variants within a certain category.

In general, there is no best storage technology and no best product for a technology. Hence, the goal is to find a storage solution that best fits to a certain application. In case of complex applications with different storage demands, even a combination of different storage solutions makes sense. Sandalage and Fowler [23] shaped the term "polyglot persistence" for such a scenario. The idea is to use the most appropriate storage technology for each specific problem. "Appropriate" might cover several aspects such as the functionality, e.g., a full power of SQL queries, but also cost aspects, latency, reliability, or security issues. For example, costs can be decreased by using slow and cheap storage whenever possible instead of fast but expensive storage. Costs can also be reduced by using a public cloud offering. However, the various and complex price schemes and underlying factors such as price/GB, data transfer or number of transactions have to be understood. However, a public cloud storage might not be the best solution for storing highly confidential data due to possible security risks or compliance with legal regulations. A hybrid cloud, i.e., combining storage in a private cloud on premises for keeping confidential data with a public cloud for non-confidential data, might be appropriate.

This paper is an extension of [14], where we suggested a multi-criteria, federation approach to cover those scenarios. The overall goal is to find an optimal data placement solution for an overall benefit. To this end, we extend our work for a cloud storage developed by the European funded VISION Cloud project [16, 17]. In contrast to conventional solutions, the overall goal of VISION Cloud is to develop next generation technologies for storing and efficiently retrieving *large* objects such as virtual machine images, high resolution imaging analysis scans, or videos.

In the following, Sect. 2 introduces the VISION Cloud software as far as it is relevant for this work, particularly, the storage architecture and the major storage interfaces. The VISION Cloud approach is called content-centric and follows the CDMI proposal [7]: There is a clear distinction between objects and containers holding these objects. The fundamental concept for retrieval is a first-class support for metadata for objects and containers. We combine the metadata concept with a federation approach [24] to cover the multi-criteria scenarios.

Therefore, we explain in Sect. 3 the original VISION Cloud federation approach that has been extended to tackle the needs of a multi-criteria storage solution. The basic VISION Cloud federation support has been used to implement two simple scenarios, which are also discussed in Sect. 3.

Afterwards, Sect. 4 introduces the new extended federation multi-criteria approach. A more flexible federation approach provides a unified and location-independent access interface, i.e., transparency for data sources, while leaving the federation participants autonomous. The presented federation approach allows taking benefit from the advantages of various storage solutions, private, on-premises and public clouds, access speed, best price offerings etc. In addition to the previous work [14], we detailed the approach and explain in Sects. 4.9 and 4.10 some further ideas about an advanced monitoring and recommendation system that better supports users. Moreover, the approach is extended to incorporate feedback from users in an appropriate manner.

In Sect. 5, we discuss related work and underline the novelty of the approach. Finally, Sect. 6 concludes the paper and mentions some future work.

2 VISION Cloud Overview

2.1 Motivation

VISION Cloud [16] was a project co-funded by the European Union for the development of cloud storage solutions with a specific focus on large objects such as videos, high resolutions diagnostics images, and virtual machine images. The overall VISION Cloud goal was to develop a storage system whereupon several domain applications are enabled to raise innovations. VISION Cloud should particularly support domains that have an increasing need for a large capacity and expect to store a vast number of large objects while also being able to retrieve stored content efficiently. Thus, typical domains targeted by VISION Cloud were health care, broadcasting and media, and IT application management, which have advanced needs for suitable storage systems due to the increasing storage capacity and large storage consumption per object and handling of very large objects. Indeed, sharing of video messages in the telecommunication domain turns into a trend since the market share of high-resolution camera equipped handsets grows [27]. Similarly, Ultra High-Definition and 4 K resolution content becomes a standard in the media domain.

In conventional approaches, large objects are stored in files and organized in a hierarchical structure. Hence, the hierarchical structure is the only mean to find a particular item by means of navigating through the hierarchy until an object is found. In fact, the hierarchy has to be set up manually and in advance without knowing all the search criteria. Hence, such a hierarchy is prone to become outdated soon or later. If the hierarchy is not understood correctly, objects might also be wrongly placed. This all leads to difficulties in finding objects in an efficient manner, especially in case of larger volumes of data. In addition, the problem arises how to maintain hierarchy changes in a distributed environment.

In order to overcome those problems, VISION Cloud pursues a metadata-centric approach and explicitly represents the type and format of the stored objects. The content-centric approach relieves a user from establishing hierarchies in order to organize a high number of storage objects. Moreover, VISION Cloud puts a lot of emphasis on flexible search options with intuitive categories for ever increasing amounts of data and acceptable access performance.

2.2 Metadata Concept

VISION Cloud relies on metadata to store and retrieve objects and the ability to perform autonomous actions on the storage node based on metadata [16]. Any content is accessed based upon using metadata. Metadata is often already available such as the author or the title of a video, a description, and ratings of users who have used the content. More technical metadata is the file format, the date of upload, and the image compression algorithm, to give some examples. Hence, metadata often exists and be easily derived when being stored, as one of the demonstrators of VISION Cloud has shown [15]. Further metadata can also be attached manually in order to classify or characterize the content. A specific VISION Cloud concept, called *storlets* [17], enhances the possibilities to derive metadata significantly. Storlets are similar to triggers in relational databases; they contain executable code and are triggered by certain events. Hence, storlets are an excellent mean to analyze and process metadata from uploaded storage objects, and to attach derived metadata to objects. For example, the storlet approach can be applied to run speech-to-text analyzers on video content in order to extract the audio track as metadata and to store the resulting text as part of the video object.

Metadata acts as search terms and improves the ability to navigate across a large number of video objects and to efficiently retrieve objects. It is important to note that metadata can be attached to objects and containers, the latter keeping several objects. Indeed, container metadata contributes to retrieving the objects inside.

To keep metadata, VISION Cloud uses a key-value objects tree in a specific storage being separated from the proper storage for the large objects. Both the storage objects and their metadata are efficiently kept in synchronization. This is an important feature and technical challenge for the internal distributed, node-based architecture targeting at horizontal scaling.

2.3 Content Centric Interface

All the applications of VISION Cloud use an interface named *Content Centric Interface* (CCI). The CCI conforms to the Cloud Data Management Interface specification [7] because CDMI provides a standard for interfaces to general object storage systems. In CDMI, storage objects are basically organized in *containers*. Containers are similar to the concept of buckets in other storage solutions such as Amazon S3. Moreover, CDMI standardizes the access to and storage of objects in a cloud by offering the typical CRUD (Create, Retrieve, Update, Delete) operations in a REST style [10] using HTTP(S).

The VISION Cloud CCI extends CDMI as, e.g., the important concept of the previously mentioned storlet is not covered by the standard. Furthermore, the major goal of VISION Cloud is not only the organization of storage objects in containers, VISION Cloud also stresses on efficiently handling and querying objects in general by using the fundamental metadata concept.

In order to create a new container for a specific tenant `MyTenant` in in the CCI, a request `PUT /CCS/MyTenant/MyContainer` has to be issued. The payload of the request contains the metadata to be attached with that container. Figure 1 shows a complete request for creating a new container:

```
PUT /CCS/MyTenant/MyContainer
  X-CDMI-Specification-Version: 1.0
  Content-Type: application/cdmi-container
  Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
  Accept: application/cdmi-container
{ metadata: { "content" : "video", "format" : "mpeg3" } }
```

Fig. 1. HTTP PUT request.

Similarly, a CCI request `PUT /CCS/MyTenant/MyContainer/MyObject` stores an object `MyObject` in the container `MyObject`. Again, the payload contains the metadata describing an object. To distinguish a container from an object, the type of data for this request is indicated by `cdmi-container` in the HTTP Content-Type header.

2.4 Vision Cloud Architectural Layering

Since the general VISION Cloud architecture was presented in detail in [16], we here concentrate on the major concepts for storage and the metadata handling. The VISION Cloud architecture is layered. The fundamental *object storage* layer provides the storage and management of metadata. This layer also handles the replication of storage objects with their metadata across the storage nodes.

Every node has a so-called *Content Centric Service* (CCS) deployed, which sits on top of the object storage layer. The CCS implements the CCI (cf. Subsect. 2.3) and provides higher-level services such as relationships between storage objects, thus enhancing the CDMI standard with additional powerful operations for a rich metadata handling. As a popular use case, having a reference of an object, an application is enabled to retrieve similar objects in the storage. For instance, the provider of some content is able to store media content in a way that is optimized for high definition displays as found in smart phones or hand-held devices. Using the CCS, applications can query for objects that are similar to the object already known. This is pretty useful if different media encoding types for different end devices come into play. The CCS determines the encoding relations by means of key-value metadata. Indeed, similarity is a metadata feature that was especially introduced to support relations between objects as metadata. Internally, the CCS decodes the relation-based query issued by an application into an internal metadata format, which does not provide the higher relation concept. A corresponding query is sent to the underlying object storage.

At the top of the layered architecture, the real applications use the CCS to access the storage.

The software of the VISION cloud storage was developed to run on common appliances. In a VISION Cloud storage system, a huge number of parallel nodes can be deployed in a data center to achieve a high scalability and load balancing. Thereupon, applications are enabled to access several nodes as far as they possess a CCS and its underlying object storage stack. The distribution of data across several storage nodes remains transparent to the application.

The CCS applies an adapter concept to allow for an integration of different storage servers. Implementing such an adapter enables one to integrate other storage implementations within the CCS: As a prerequisite, any adapter has to take care of handling metadata besides the real object storage. That is, the CCS can work with other storage systems as well if they are capable of storing metadata attached to objects and containers. The open source CouchDB document database was integrated as one candidate into the CCS as part of the VISION Cloud project in order to prove applicability.

The CCS connects to a storage server's IP and port number by means of a specific storage adapter. Behind the IP, there might be a single storage server. But it is also possible that the IP address points to a load balancer with a cluster implementation behind. From a technical point of view, there is no need for the object storage and the CCS to reside on the same machine or node. The request handling can be easily deployed on the storage node as a module since it does not need to support sessions. Applications are allowed to bypass the CCS, thus accessing the object storage directly. In that case, it is however not possible to use the advanced CCS metadata handling capabilities. Furthermore, the CCS is capable of avoiding node management functionality and keeping track of the current status of object nodes. This all supports a general scale-out of VISION Cloud storage nodes.

The basic setup of VISION Cloud nodes deploys a CCS on every storage node, which is a (potentially virtual) server running a VISION Cloud node software stack. Hence, response times are reduced since connections between different network nodes are avoided.

3 Federations in VISION Cloud

3.1 Fundamental Support for Federations in VISION Cloud

In general, a federation has the goal to yield an access interface so that a transparency of data sources is achieved while at the same time let the federation participants stay autonomous. A Cloud federation should thus abstract from different storage clouds of different provisioning modes. Clients are offered by the federation a unified view of storage and data services across several systems and providers.

In particular, a federation has to tackle heterogeneity of the involved data sources. There are several types of heterogeneity. At first, each cloud provider offers proprietary Application Programming Interfaces (APIs) and management concepts. In fact, industry specifications such as CDMI [7] are a step towards homogenization. Furthermore, another type of heterogeneity to be handled by the federation is that cloud storage providers apply heterogeneous data models.

The implementation of the CCS of VISION Cloud already provides features for handling heterogeneity by means of an adapter concept. Each adapter abstracts from an underlying heterogeneous source by offering a common CCS interface. An instance of a VISION Cloud CCS sits on top of each storage system. CCS adapters are already implemented for the proprietary VISION Cloud storage service and CouchDB. Further adapters can be implemented for other storage systems. Consequently, the CCS approach allows for integrating multiple storage system types of various cloud providers.

Each CCS uses an IP connection to communicate with a storage system. In particular, a CCS can be based upon a load balancer that itself possess several homogeneous storage servers beneath and works as a cluster solution thereby bridging the load balancer and the client. Hence, all scalability, elasticity, replication, duplication, and partitioning is delegated to the storage system. Indeed, current storage system types usually have a built-in cluster implementation already. For example, the open source project MongoDB possesses several strategies for deploying clusters of MongoDB instances. And CouchDB provides an elastic cluster implementation called BigCouch. These cloud systems are able to deal with thousands of servers located in many data centers around the world thereby serving millions of customers. Hence, the complexity of CCS is drastically reduced by not re-implementing such distribution, scalability, load balancing, and elasticity features.

Moreover, VISION Cloud offers some fundamental federation functionality for distributing requests across multiple storage nodes. Using the metadata approach, the CCS is able to route storage requests to different storage services similar to a load balancer. Although not really representing a load balancer, the role of the CCS in the software stack is useful for similar purposes. For example, the CCS can distribute requests over different storage nodes depending on quality characteristics, matching the provided capabilities of the underlying storage clouds.

Furthermore, VISION Cloud integrates fine granular access control lists (ACLs), which can be attached to tenants, containers, and objects.

These basic VISION Cloud federation features have been used in two major federation scenarios, which are presented in the following subsections.

3.2 Using the Federation Concept for On-Boarding

According to [11], vendor lock-in is the second obstacle among the top ten for growth in cloud computing. A specific on-boarding federation supports clients to become independent of a particular cloud storage provider: On-boarding means that data can be transferred between clouds without operational downtime [26]. Hence, data can be accessed while the migration migrated from one storage system to another is in progress.

In a first step, an administrator has to set up an on-boarding federation between the two involved storage systems, the source and the target. To this end, the administrator sends a “federation configuration” to a federation service. This configuration specifies the access information of the remote (i.e., the source) container including credentials. Figure 2 describes the payload of a typical federation configuration.

```
"federationinfo": {
  // information about remote cloud
  "eu.visioncloud.federation.status": "0",
  "eu.visioncloud.federation.job_start_time":
    "1381393258.3",
  "eu.visioncloud.federation.remote_cloud_type": "S3",
  "eu.visioncloud.federation.remote_container_name":
    "example_S3_bucket",
  "eu.visioncloud.federation.remote_region": "EU_WEST",
  "eu.visioncloud.federation.type": "sharding",
  "eu.visioncloud.federation.is_active": "true",
  "eu.visioncloud.federation.local_cloud_port": "80",
  // credentials to access remote cloud
  "eu.visioncloud.federation.remote_s3_access_key":
    "AFAIK3344key",
  "eu.visioncloud.federation.remote_s3_secret_key":
    "TGIF5566key",
  "eu.visioncloud.federation.status_time":
    "1381393337.72"
}
```

Fig. 2. Sample payload.

The federation service is deployed along with the CCS and offers typical operations to configure and handle federations by means of REST-ful services. A federation instance can be created with a PUT request passing an id in the Uniform Resource Identifier (URI) and the federation configuration in the body. A GET-request returns the federation progress or statistical data for a particular federation instance. DELETE removes a federation configuration. More details can be found in the VISION Cloud project deliverable [28].

Having configured a federation for the source and target systems, the transfer of objects of federated containers can be initiated. While data is being transferred to the target container in the background, clients can still request the contents of the container even if the transfer of that container has not yet finished: All objects of all containers are available. Containers that have not been completely transferred will be fetched from the remote source by using the federation information. The federation service possesses an on-boarding handler that intercepts GET-requests from the client and redirects them to the remote system for non-transferred containers. For transferring data, the information is used by the target container to access the remote server and to fetch data from the remote to the target server. A federation works on a container level. Moreover, the handler is responsible for scheduling the background jobs for transferring data.

3.3 Hybrid Cloud Federations

In another scenario, a federation is used to set up a hybrid cloud [13], consisting of a public and a private cloud. Hybrid clouds are useful if both critical or privacy data and non-critical data has to be stored. For example, regulatory certifications or legal restrictions forces one to store material that is legally relevant or subject to possible confiscation on premises, on private servers. Since private servers are more expensive, it is not reasonable to keep non-critical data on the same private server. Instead, more efficient cloud offerings from external providers, which are cheaper and offer better extensibility, should be used. The idea is to split a "logical" container across physical public and private cloud containers and to use metadata in order to control the location of objects. That is, a PUT request for storing objects can obtain some metadata that specifies whether the data is confidential or not.

As a prerequisite, every logical container has to know its physical locations. This can be done by means of an administrative PUT request with the payload of Fig. 3, which has to be sent to the federation service of vision-tb-1. Internally, the second cloud receives an "inverted" payload. Afterwards, the two cloud containers know the locations of each other.

```
https://vision-tb-1.myserver.net:8080/MyTenant/vision1
{
  "target_cloud_url" : "vision-tb-2.myserver.net",
  "target_cloud_port" : "8080",
  "target_container_name" : "logicalContainer",
  "local_container_name" : "logicalContainer",
  "local_cloud_url" : "vision-tb-1.myserver.net",
  "local_cloud_port" : "8080",
  "type" : "sharding",
  "private_cloud" : "vision1",
  "public_cloud" : "vision2"
}
```

Fig. 3. Federation payload for configuration.

The configuration in Fig. 3 specifies the private and public cloud types, URLs, users, authorization information etc. The `private/public_cloud` specification determines that the cloud server `vision-tb-1` will contain the private container and `vision-tb-2` the public one. Every container requires such a specification if being split into a private and public part. Hence, the federation occurs at container level.

The distribution of data across a private and public part is transparent to clients. A client is completely unaware of the hybrid cloud setup and does not know where the data is kept. During the creation of objects, metadata is used to determine data confidentiality by a metadata item `confidential:"true"|"false„` for instance in the request presented in Fig. 4.

```
PUT vision-tb-1.cloudapp.net:8080
                   /CCS/siemens/logicalContainer/newObject
{ "confidential" : "true" }
```

Fig. 4. PUT request for storing confidential data.

To enable a hybrid scenario, a ShardService had to be introduced as part of the CCS. The CCS handles the PUT request by checking the metadata of the object and delegates the request to the public or private cloud accordingly. Indeed, the CCS provides all the content-centric storage functions. The ShardService implements a reduced CDMI interface and has the key role in hybrid scenarios by intercepting each request to a CCS and to decide where to forward the request. Internally, the connection information is determined for both clouds by using the hybrid cloud setup. In case of `"confidential":"true"„` the ShardService decides to store `newObject` in the private cloud. Thereby, the ShardService uses the ability of the CCS to connect to multiple object storage nodes at the same time. A client is able to connect to any Cloud server and does not need to know the specific location of a container.

Similarly, clients obtain a unified view of the data, no matter whether located in the private or public. Hence, a GET request to a federated container is sent to all clouds participating in the federation unless confidentiality is part of the query and the server is thus known. The results of the individual requests for the private and public containers are collected, and the combined result is returned to the client.

Every operation can be sent to the public and private cloud store in the federation, i.e., each cloud can accept requests and is able to answer. There is no additional interface to which object creation operations and queries need to be submitted.

4 Multi-criteria Storage Federations

The potential of the metadata approach for federation has already been illustrated in Sects. 3.2 and 3.3. Indeed, metadata is able to keep information that is then used to determine the storage location. The approaches are enhanced in this section to achieve a better flexibility of specifying and selecting cloud storage systems in order to take full benefit from their particular capabilities.

4.1 Basic Idea

The overall approach provides means to let a user influence the storage of data. For example, putting higher demands on privacy lets objects be stored in locally hosted cloud storage systems, while less critical data is stored in public cloud offerings. Furthermore, the approach benefits from special features of cloud storage providers such as reduced resilience: It is reasonable to store data that is only temporarily required in some cheaper cloud storage with reduced redundancy. The pursued approach is a so-called multi-criteria storage. The term multi-criteria thereby underlines the ability of the approach to organize the storage based upon multiple user criteria. Relying on the VISION Cloud federation concept [26] to define a setup of heterogeneous clouds, it becomes possible to involve several cloud storage systems each possessing various properties and benefits. Moreover, the metadata processing features of VISION Cloud build an important component of the multi-criteria approach. In fact, the idea relies on applying metadata to let users ask for individual storage criteria for both containers and objects.

The VISION cloud deployment sets up a federation admin service (including an interface) for each cloud storage participating in the federation. Every cloud storage node offers the same service interfaces.

Let us assume several cloud storage systems (CSs) named CS1, CS2, CS3 etc. Each of these CSs should possess dedicated properties with respect to availability SLAs, privacy issues, and access speed. Indeed, the CS properties follow the resource model (cf. Sect. 4.2).

The definition of properties for a particular CS requires an administrative PUT request to the federation admin service of a VISION Cloud. In contrast to the federation scenarios in Section [3], a different payload has to describe the capabilities of every CS – for the whole CS, not for particular containers or objects. Each member of the federation receives the payload to become aware of the CS and its particular properties.

The federation principle again benefits from the metadata approach and let CCI operations thus work at a higher level of abstraction by hiding the underlying storage systems and their interfaces. The existing CCI operations behave as described in the following subsections.

The Content Centric Service (CCS) component of VISION Cloud [15] has to process the metadata with the additional task of satisfying those requirements.

4.1.1 Creation of a New Container

A user has to issue a PUT request that contains metadata according to a requirements model (cf. Sect. 4.3), thus specifying the desired properties. The PUT request can be sent to the CCI of any CSi. Every container maintains metadata about its own properties and the mapping to its list of further containers in other storage systems. The CSi analyzes the request and is able to decide where to create the container in order to fulfill the specified requirements at best. In some cases, several CSs might be chosen, e.g., to achieve higher availability by keeping data redundantly. It should be noted that other CSs are involved in satisfying the storage request, if necessary. Every CSi is capable to receive and answer requests, even if the container will finally be directed to a different CS.

4.1.2 Creation of a New Object

Again, creating a new object in a container requires a PUT request with user requirements to be sent to the CCI of any CSi. As a general design decision, storage criteria can be attached to both containers and objects. Thus, we are able to support advanced use cases where a per-object decision is useful. For example, a "logical" container might keep images, some of them are confidential (e.g., anonymous versions of the same press images), while others are not confidential (e.g., non-anonymous press images). Accordingly, different CSi are used for storage. Some images could be easily recovered (i.e., thumbnails of original images) and thus do not require high reliability, while other objects are relevant on the long term. Or some objects might be rarely accessed while others are used frequently. In any case, the CSi takes the decision to choose the appropriate CS(s). Indeed, the container criteria lead to a default storage location for all its objects, which can be overridden on a per-object basis.

4.1.3 Retrieval of Objects

If a user wants to retrieve a particular object in a container, each CS must be asked unless the criterion determines one or several CSs. Hence, every CSi must be able to determine the possible object locations by means of using the metadata. The query is performed in all the CSs in parallel.

4.2 Resource Model

Our major goal is to use a federation approach in order to place data optimally on different storage systems according to user-defined functional and non-functional requirements. In a first step, the physical resources of the underlying storage solutions need to be precisely described. To this end, we can benefit from the VISION Cloud management models [12], especially the resource model. The resource model provides a framework for specifying all the various features of storage systems in a uniform way. For example, the resource model specifies several technical properties of the federation participants:

- the cloud provider (e.g., Amazon) the type of store (e.g., S3 Blob Store);
- public and private cloud offerings;
- replication factor if the storage internally replicates data;
- technical properties about disk accesses (e.g., high I/O SSD storage);
- latency of data center for locations;
- the structure of the pricing scheme for storage, usually based upon cost on per GB/month, incoming and outgoing data transfer, the number of transactions and others.

The model follows a hierarchical approach from cloud to node level that allows attributes to be defined on different levels.

4.3 Requirements Model

A requirements model is required to properly capture and structure the requirements emerging from application attributes modeling and user needs. Users should be enabled to specify *non-technical* requirements in order to relieve them from technical aspects. Again, we rely on the VISION Cloud User Requirements and Resource Requirements Model as part of the developed management models [12]. The requirements model mainly covers the user requirements in a formalized way. Requirements are high-level criteria and mainly characterize the application data and needs for an appropriate storage service. Indeed, those requirements have to be translated to the more low-level storage characteristics given by the resource model somehow.

The following are typical criteria a user can use to specify for creating containers or objects:

- storage cost (at certain granularities);
- degree of confidentiality;
- level of reliability (possibly requesting replication);
- access speed (requiring fast disk access and low latency);

Further criteria can be found in [12]. In fact, these criteria have an indirect impact on the placement of objects and/or containers in a storage system. The requirement model allows a user to specify his demands in terms he/she is familiar with, whereupon the infrastructure is enabled to define a specific strategy regarding placement of data to satisfy these demands.

4.4 Administrative Tasks to Set Up a Mapping

A user uses the requirements model to define the usage-related criteria – independently of the physical resource model of CSs. In fact, these criteria have to be somehow mapped to the physical storage properties of the resource model. This enables mapping high-level user requirements, described as metadata, to low level resource requirements and thus forms the basis to find an appropriate storage system fulfilling the specified user criteria. The important new task is to map the specified user criteria to the technical storage system parameters in such a way that a storage system fitting best to the criteria will be found. This task is done within the federation service by an appropriate mapping approach.

Similar to the general approach of VISION Cloud, an administrator of the system should perform the configuration of the system using the same REST/JSON-based approach, which is basically defined by CDMI. The already available VISION Cloud federation service provides some generic configuration REST calls that can easily be extended by a PUT operation to let an administrator submit configurations.

The basis for determining appropriate Storage Systems (CSs) for given user requirements is a list of storage systems with their properties. Hence, in a first administrative step, each storage system is described by certain categories according to the resource model: public/private, a redundancy factor, its location, access speed, latency etc.

Table 1 gives an example of such a specification in a tabular manner. Each row specifies a CS the properties of which are defined in various columns such as the type, provider, typical access speed, redundancy factor, public vs. private Cloud etc.

Similar to Fig. 4, the payload now obtains the access properties (url, port etc.) and also the properties of a participating CS, as shown in Fig. 5.

Table 1. Table RESOURCE – storage system properties [14].

Storage system	Type	Public	Provider	Redundancy factor	Access speed	Location
CS1	SSD drive	No	Samsung	1	1 ms	Local
CS2	Vision store	No	Vision	2	10 ms	Local
CS3	Blob store	Yes	Amazon	3	5 ms	Amsterdam
CS4	Table Store	Yes	Azure	3	20 ms	Dublin
CS5	Blob store	Yes	Azure	3	8 ms	Dublin

```
https://vision-tb-1.myserver.net:8080/MyTenant/vision1
{
  "storage system" : "CS1",
  "url" : "vision-tb-1.myserver.net",
  "port" : "8080",
  "type" : "SSD Drive",
  "provider" : "Samsung",
  "redundancy factor" : "1",
  "access time" : { "10", "ms"},
  "location": "local",
  "public_cloud" : "no"
}
```

Fig. 5. Extended federation payload.

4.5 User Requests

Using CDMI, a user can send a request to create a container or object with associated metadata requirements, i.e., the user criteria according to the requirements model, to the CCS. Such a request resembles Fig. 4, but now specifies the user criteria as metadata together with a percentage ranking, as Fig. 6 shows.

```
PUT vision-tb-1.cloudapp.net:8080
                 /CCS/siemens/logicalContainer/newObject
{
  "confidentiality" : { "50", "%" },
  "reliability"     : { "20", "%" },
  "cost savings"    : { "30", "%" }
}
```

Fig. 6. PUT request for storing an object with user criteria.

In this example, a user specifies a weight of 50% for the dimension “Confidentiality” and weights of 30% for the dimension “Reliability” and 20% for the dimension “Cost Savings”.

The CCS contacts the federation service to ask for the relevant federation information. The federation returns the referring storage location(s) accordingly.

4.6 Technical Details

In knowledge-based systems, the problem of finding a recommendation is usually formulated as a pair (R, E). Here, R corresponds to the set of user requirements (according to the requirements model) while E is the set of elements that form the knowledge base. That is, the features of the resource model, i.e., the entries in Table 1, are the elements E.

The solution for a recommendation task (R, E) is a set $S \subseteq E$ that has to satisfy the following condition:

$$\forall e_i \in S : e_i \in \sigma_R(E)$$

$\sigma_R(E)$ should here denote a selection restricting the elements in E to those that satisfy R. For instance, suppose the user requirements are defined by the concrete set:

$$R = \{r_1 : \text{access speed} \leq 15,\ r_2 : \text{public} = \text{yes}\}$$

Obviously CS5 is the only storage system that satisfies such a requirement R.

Table 2. Mapping table MAP [14].

	Confidentiality	Reliability	Cost savings	…
public=yes	1	5	8	
public=no	9	2	2	
redundancy factor=1		1	9	
redundancy factor=2		3	5	
redundancy factor > 2		9	1	
provider=Amazon		7	5	
provider=Azure		7	4	
provider=Samsung		5	3	
provider=Vision		8	5	
access speed>=30 ms			8	
access speed<30 ms			6	
access speed<20 ms			4	
access speed<10 ms			1	
…				

The missing step is the procedure for recommending appropriate storage systems. In order to recommend appropriate systems for the user storage request, a Multi-Attribute Utility Theory (MAUT) is used. In general, a MAUT schema allows assessing and ranking a user's resource requirements according to the dimensions of a particular interest. The dimensions of interest are the user requirements such as reliability or cost savings. To set up a MAUT schema, the individual property values of the storage systems are related to the dimensions of interest with a certain ranking ranging from 1 to 10; a value of 10 means highly relevant. Consequently, each entry of the MAUT schema contains a value that defines the relevance for the related dimension by associating a certain weight. The larger the value is, the more the property contributes to the dimension of interest.

Table 2 presents a sample MAUT schema. Each row represents an individual property value, i.e., each facet of the user requirements model like public is split into several rows according to the number of values. In order not to flood the table, comparisons like '`redundancy factor>2`' are useful, especially if a finer classification is not reasonable.

The content of Table 2 should be understood as follows. The first line ('`public=yes`') specifies that a public infrastructure as a resource requirement contributes to

- `confidentiality` (second column) with a quite low weight of 1 because a public Cloud infrastructure cannot be considered confidential;
- `reliability` with a medium weight of 5;
- `cost savings` with a high weight of 8
- etc.

Similarly, '`private = no`' contributes to

- `confidentiality` with a weight of 9;
- `reliability` with a weight of 2;
- `cost savings` with a weight of 2
- etc.

Analogously, the `redundancy factor`, more precisely the corresponding values or ranges 1, 2 and >2, has an impact on the reliability (increasing with the factor), to the cost savings (decreasingly) etc., but none on confidentiality (empty cells).

4.7 Determining Recommendations

Based upon this data, the recommendation takes place for a given user request like the one in Fig. 6. Indeed, a request U formulates a user's request including a ranking. Finding adequate storage systems implies that these requirements have to be satisfied.

To formalize the approach, we introduce the following notions:

- `WEIGHT(U,d)` corresponds to a user request and should specify the percentage for a request U for each dimension d of interest in a request. For example: `WEIGHT(<request of Fig. 6>,"confidentiality") = 50`
- `RESOURCE`: *`StorageSystems`* × *`Properties`* → *`Values`* is function reflecting the contents of Table 1. As an example, `RESOURCE("CS1", "public")` calculates to the value `"no"`.
- `MAP`: *`Properties`* × *`Values`* × `Dimensions` → `Int`

represents the MAUT mapping table. For example, `MAP("public","yes","Reliability")` refers to the first line in Table 2 and computes to the value 5.

The following formula then computes the relevance of a storage system in a weighted manner using those functions:

$$\mathtt{Relevance}(\mathtt{U},\mathtt{CS_i}) = \sum\nolimits_{\mathtt{d}\in\mathtt{Dimensions}}\Big(\sum\nolimits_{\mathtt{p}\in\mathtt{Properties}} \mathtt{MAP}(\mathtt{p},\mathtt{RESOURCE}(\mathtt{CS_i},\mathtt{p}),\mathtt{d}) \times \mathtt{WEIGHT}(\mathtt{U},\mathtt{d})\Big)$$

Using this formula, for each CSi a value for each dimension is calculated. Table 3 illustrates the calculation for our example. The line for CS1 should be understood as follows: `Confidentiality` is only determined by '`public=yes`' and '`public=no`'. That is CS1 and CS2 obtain a value of 9 because of '`public=no`', while the others have a value of 1. `Reliability` is affected by '`public=`', '`redundancy factor=`', and '`provider=`'. Since CS1 has the properties '`public=no`', '`redundancy factor=1`', and '`provider=Samsung`', the value computes to 2 + 1 + 5 according to Table 2.

Table 3. Sample calculation [14].

Storage system	Confidentiality [50%]	Reliability [20%]	Cost savings [30%]	Overall result
CS1	9	2 + 1 + 5	2 + 9 + 3 + 1	10.6
CS2	9	2 + 3 + 8	2 + 5 + 5 + 4	11.9
CS3	1	5 + 9 + 7	8 + 1 + 5 + 1	9.2
CS4	1	5 + 9 + 7	8 + 1 + 4 + 6	10.4
CS5	1	8 + 1 + 1	8 + 1 + 4 + 1	8.9

The last column in Table 3 takes the values and weights them with the relevance percentage for each domain of interest. The storage CSi with the highest overall value is suited best to satisfy the demands. That is, CS2 having a value of 11.9 is best suited for the given set of requirements while CS5 with a value of 8.9 is the worst choice.

4.8 Properties of the Approach

In spite of the examples being kept intentionally simple, the potential for users is high. For example, the price schemes of Cloud providers are quite complex, taking into account various factors such as type of Virtual Machine, in/outgoing data transfer, storage capacity (provisioned or used), and transaction rates. In fact, these all are rather technical properties of a (Cloud) storage system (in the sense of a resource model, cf. Sect. 4.2), which can be rated and mapped to the "Cost Savings" user criterion according to what aspect dominates the costs. Due to our knowledge, complex price schemes cause problems for users to estimate the overall costs.

In general, satisfying a set of requirements might lead to a conflict solving all the requirements. For example, a request

$$\mathtt{R} = \{\mathtt{r}_1 : \mathtt{public} = \mathtt{no}, \mathtt{r}_2 : \mathtt{redundancy\,level} > = 2,$$
$$\mathtt{r}_3 : \mathtt{access\,speed} \leq 10\}$$

does not possess a solution for the data in Table 1, i.e., $\sigma_{\mathtt{R}}(\mathtt{E}) = \emptyset$, or in other words, R raises a conflict. A relaxation R_{relax} of R, e.g., abstaining from `r1`, could lead to a solution. Indeed, such a relaxation should be minimal in the sense that so subset $R'_{relax} \subseteq R_{relax}$ exists with $\sigma_{\mathtt{R'Relax}}(\mathtt{E}) \neq \emptyset$. There is a typical problem of finding those relaxations in an efficient manner.

The advantage of our approach lies in the fact that no conflicts can occur compared to [29]. Indeed, conflicts might occur at the lower, physical level of resources, as described above. However, users are not directly formulation requirements at that lower level. Instead, more abstract requests are formulated according to the requirements model. Even if a request would equally rate all requirements like confidentiality, reliability, and cost savings with 33.3% or any other percentage, no conflict could arise: There will always be a compromise found. Using percentages, the user is able to control some preferences. Hence, no conflict detection is required as in other recommendation systems.

As a disadvantage, the user is unable to declare a physical property such as `private=yes` as an absolute must.

4.9 Monitoring and Recommendation System

There are two challenges related to the multi-criteria approach:

1. The approach relies on the correctness of the storage properties as being fed into the system. However, the specified values might be wrong or might change over time. As an example, popularity of videos might change over time leading to different optimal storage characteristics.
2. The user specifies his demands on storage, but he might have a different perception about his usage scenarios.

To tackle both points, we have set up an advanced monitoring and recommendation system. The task of the monitoring system is to collect all metrics that can be used to optimize the placement of data, both on the application as well as on the hardware level.

Interesting metrics include:

- number of accesses to an object;
- access speed;
- geographic location of requests;
- downtime (recognized by failures returned by a storage system);
- costs (either calculated online according to price schemes or using the Cloud provider's bill).

4.9.1 Monitoring System Architecture

The metrics described above should be collected from the different sources that can be physically distributed and propagated to a higher level where decision making components can process them. In order to do so effectively, it is best if information is aggregated as much as possible on the lowest level. Therefore the developed monitoring system follows a distributed model. Similar to the CSS, a monitoring instance runs on every node of the VISION Cloud. The instance is comprised of:

- collectors, for collecting hardware as well as application level metrics (such as access requests to the CCS component),
- a rule engine that is used to evaluate every event received and to trigger its preprocessing or immediate dispatching to consumers,
- an aggregator that can aggregate and process events over specific time-windows, and
- a dispatcher that is responsible for communicating relevant events to consumers using a high performance messaging library (ZeroMQ [30]).

The rules format is consisted of three sections:

- the aggregation period
- a “when” section specifying basic Boolean operation on the fields of the events, and
- a “then” section specifying the action to be taken in the form of a function.

An example of a rule that sums all the bytes read from objects belonging to a specific tenant over a 10 min span can thus be:

```
agg_period=10
when
  (e.op = 'read') AND (e.tenant = 'sie')
then
  SUM e.bytes
```

By aggregating this information on the lowest possible level we avoid unnecessary network communication at the expense of processing power. The aggregated information from multiple nodes can in turn be collected on a higher level where complex events can be produced and consumed by the recommendation system.

When an external system is federated with a VISION Cloud system, a dedicated component uses the APIs exposed by the federated system to collect relevant information and propagate this to the local representative of the monitoring system. Information is thus collected in an efficient manner from all the nodes of the system and then stored in a distributed database. This information can be used in different ways.

4.9.2 Usage of the Monitoring and Recommendation System

Firstly, collecting this information and analyzing it enables the system to adjust the storage properties to reality. Such an approach relieves the user from explicitly giving feedback. Due to our experience, a user often feels unable (and/or annoyed) to judge the quality of a service, i.e., how good his storage request is satisfied. Hence, an automatic approach seems to be more appropriate.

Secondly, the user requests are also checked by the monitoring system and recommendations can be made. For example, a user request might have given for a specific container or dedicated objects fast access a higher priority than costs for the high quality storage. However, the monitoring component might detect that these objects are accessed only sporadically. Obviously, the user had a wrong understanding of his accesses at the time s/he specified the request properties. The recommendation system is able to help in such a situation by notifying "your original storage request asked for fast access for the following objects, however, you are accessing those objects only once a week. We recommend moving the objects to a cheaper location by changing your storage request properties to the following: …" even if cost savings did not have a high priority in the original storage request. Similarly, assume cost savings are relevant for the user. The recommendation system is able to detect cheaper storage solutions by monitoring accesses and verifying the other specified storage parameters.

Although the idea is quite straightforward, we detected some challenges during the implementation. At first, any implicit modification of storage parameters made by the monitoring component – requires a re-location, not only of the directly affected users' objects but also all the other objects and containers. Indeed, the location of other objects also relies on valid storage parameters. Such a re-location is required for manually adjusted (by the administrator) because of known changes as well. Also, adding a new storage system node to the federation might have a similar impact on the distribution of objects to storage systems. Thus, such changes might result in a chain of transfer operations that would require prediction of the resulting load and effort put onto the federation. There are some further negative implications. Any movement of data might cause additional cost depending on the cloud storage solution used and presumably affect the availability of data during the relocation process. In particular, public Clouds charge for in- and outgoing data transfer, which could sum up to enormous cost in case of larger data volumes. Consequently, the cost saving criterion has to be re-considered in more detail in order to reveal the transfer costs. The same holds for user recommendations if cheaper storage locations are suggested.

4.10 Automatic Adjustment of Mappings

The presented approach not only recommends a storage system for particular requirements, but also immediately uses the best system – as decided by the recommendation component – for storing containers and objects. This works fine if the mapping table (cf. Table 2) is correct.

We are currently considering an alternative to let users affect the taken decision for a storage system. Consequently, the recommendation system only *suggests* a storage system, which can afterwards be *overridden* by a user by choosing another system while still keeping the original requirements. Since the user has obviously found a more suitable storage system for his/her needs, the mapping table does not seem to be optimal.

To improve the situation, we pursue the goal is to automatically adapt the mapping table in such a way that the selection preferred by the user would have been obtained. To this end, we applied a heuristic in a first version that is currently under evaluation and will be improved in the future.

Let us continue the example in Table 3 and let us assume that the user prefers CS1 instead of CS2 as suggested by the recommendation component. In a first step, the heuristics computes the difference of results

$$\texttt{Delta(CS2,CS1)} = \texttt{Result(CS2)} - \texttt{Result(CS1)} = 1.3$$

between both storage systems. If the difference is larger than a configurable threshold, an adjustment does not seem to be reasonable and will thus be renounced.

Otherwise, a second step finds the differences of the storage systems in the REOSURCE table (Table 1). In case of CS1 and CS2, both systems possess the same properties `Public=no` and `Location=local`. It is not reasonable to change these two so that only the remaining properties are potential candidates for adjustments. The number of candidates is thus limited.

In principle, we have to find new values for those remaining candidates in the mapping table such that $\texttt{Result}_{\texttt{new}}(\texttt{CS2}) - \texttt{Result}_{\texttt{new}}(\texttt{CS1}) < 0$, but as small as possible. Thereby, $\texttt{Result}_{\texttt{new}}$ should refer to the new calculation of the result according to the adjusted values. This is a solvable problem.

In our example, we obtain the following candidates for adjustments as highlighted in bold type in Table 4.

Table 4. Candidates (Color table online)

Storage System	**Confidentiality [50%]**	**Reliability [20%]**	**Cost Savings [30%]**	**Overall Result**
CS1	9	2 + **1** + **5**	2 + **9** + **3** + **1**	10.6
CS2	9	2 + **3** + **8**	2 + **5** + **5** + **4**	11.9

The bold numbers in Reliability refer to redundancy factor and provider, while the bold numbers in Cost Savings are calculated based upon redundancy factor, provider, and access speed. Concerning Reliability, any increment by 1 in CS1 (or decrement by 1 in CS2, respectively) reduces the difference `DELTA` by 0.2, while any increment by 1 within Cost Savings reduces the difference by 0.3. Obviously, we have the following options to bridge the gap of `DELTA = 1.3`:

(a) Reliability + 7 => 1.4
(b) Reliability + 6, Cost Savings + 1 => 1.5
(c) Reliability + 4, Cost Savings + 2 => 1.4
(d) Reliability + 3, Cost Savings + 3 => 1.5
(e) Reliability + 1, Cost Savings + 4 => 1.2
(f) Cost Savings + 5 => 1.5

Please note that Reliability + 5, Cost Savings + 2 is not an option as c) requires lower values, and similar for Reliability + 2, Cost Savings + 4. Within each group, there are again several options to realize the +x. For instance, (Reliability + 1) as part of (e) can be achieved for CS1either by

- increasing the value for “redundancy factor=1” by 1 or
- increasing the value for “provider=Samsung” by 1

in order to provide a better ranking for CS1, or the other way round by decreasing the values analogously for redundancy factor=2 and provider=Vision for CS2. Similarly, combinations for cost savings can be derived.

In a final validation step, the combinations are used to calculate for the most recent recommendations in order to check what combination of new values would have falsified previous recommendations done in the past. Only if none of the recent recommendations suggests a different storage solution, the adjustment will become effective.

This approach seems to be promising despite being simple to realize in an efficient manner.

5 Related Work

Even if cloud federation is a research topic, the basic concepts and architectures of data storage federations have already been discussed many years ago within the area of federated database management systems. Sheth and Larson [24] defined a federated database system as a “collection of cooperating but autonomous component database systems” including a “software that provides controlled and coordinated manipulation of the component database system”. They described a five-layer reference architecture for federated database systems. According to the definition, the federated database layer sits on top of the contributing component database systems. One possible characterization of federated systems can be done according to the dimensions of distribution, heterogeneity, and autonomy. One can also differentiate between tightly coupled systems (where administrators create and maintain a federation in advance) and loosely coupled systems (where users create and maintain a federation on the fly).

The MetaStorage system [3] seems to be most comparable to our approach since it is a federated cloud storage system that is able to integrate different cloud storage providers. MetaStorage uses a distributed hash table service to replicate data over diverse storage services by providing a unified view between the participating storage services or nodes.

Based upon Google App Engine, Bunch et al. [6] present a unified API to several data stores of different open source distributed database technologies. Such a unified API represents a fundamental building block for working with cloud storage as well as on-premises NoSQL database servers. However, the implementation provides access only to a single storage system at a time. Hence compared to our CCS solution, the main focus is on portability and not on a federated access.

Redundant Array of Cloud Storage (RACS) is a cloud storage system proposed by Abu Libdeh et al. [1]. RACS can be seen as a proxy tier on top of several cloud storage providers, and offers a concurrent use of different storage providers or systems. Adapters for three different storage interfaces are discussed in the paper, however, the approach can be expanded to further storage interfaces. Using erasure coding and distribution, the contents of a single PUT request are split into parts and distributed over the participating storage providers similar to a RAID system. Any operation has to wait until the slowest provider has completed the request. While this approach splits data across storage systems, our approach routes a PUT request to the best suited storage system.

Brantner et al. [4] build a database system on top of Amazon's S3 cloud storage with the intention to include support for multiple cloud storage providers in the future. In fact, Amazon S3 was also one of storage layer options that VISION supports.

There is a lot of ongoing work in the area of multicloud APIs and libraries. Their goal is also to enable a unified access to multiple different cloud storage systems. Among them, Apache Libcloud [2], Smestorage [25] and Deltacloud [9] should be mentioned. They provide unified access to different storage systems, and protect the user from API changes. However, only basic CRUD methods are supported, mostly lacking of query functionality referring to the metadata. Moreover, they have administration features such as stopping and running storage instances.

Further notable approaches can be found in the area of Content Delivery Networks (CDN). A content delivery network consists of a network of servers around the world which maintain copies of the same, merely static data. When a user accesses the storage, the CDN infrastructure delivers the website from the closest servers. However, Broberg et al. [5] state that storage providers have emerged recently as a genuine alternative to CDNs. Moreover, they propose a system called Meta CDN that makes use of several cloud storage providers. Hence, this does not represent a storage system in the sense of database federations. As CDNs in general, Meta CDN mostly focuses on read performance and neglects write performance and further criteria as well. Anyway, the system provides cheaper solution by using cloud storage.

There are also a couple of hybrid cloud solutions in the literature. Most of them focus on transferring data from private to public, not providing a unified view to hybrid storages. For instance, Nasuni [19] is a form of network attached storage, which moves the user's on-premise data to a cloud storage provider. Hence, this hybrid cloud approach gathers and encrypts the data from on-premise storage, afterwards sending the encrypted data to a public cloud at either Microsoft Azure or Amazon Web Services. Furthermore, a user has the option to distribute data over multiple stores. Compared to our multilevel approach, Nasuni is a migration approach that eventually moved data to the public cloud. Another example is Nimbula [20], which provides a service allowing the migration of existing private cloud applications to the public cloud using an API that permits the management of all resources. CloudSwitch [8] has also developed a hybrid cloud that allows an application to migrate its data to a public cloud.

A hybrid cloud option has been developed by Nirvanix [21]. Nirvanix offers a private cloud on premises to their customers, and enables data transfer to the public Nirvanix Cloud Storage Network. However, other public cloud platforms than Nirvanix are not supported. Due to our adapter approach, VISION is not limited to a specific public cloud service.

As a general observation that can be made, most commercial federated and hybrid cloud storage solutions do provide a range of offerings to satisfy various customers demands, but pose the risk of a vendor lock-in due to the use of own infrastructure.

6 Conclusions

Nowadays, several storage solutions are available, no matter whether traditional relational databases, deployed on premises or in a Cloud, other Cloud offerings like blob or table stores, or more recent technologies such as NoSQL databases [22]. Obviously, each technology and even more each individual product has virtues of its own, let it be specific functionality like advanced redundancy concepts or other aspects like costs or performance.

This paper tries to tackle such heterogeneity by a multi-criteria federation approach. The approach allows using several storage technologies with different properties for different purposes in parallel, particularly finding an appropriate storage solution according to individual requirements. A federation concept is used to offer a uniform access interface to several heterogeneous systems, thereby leaving the participating systems widely autonomous [24].

The presented approach is based upon the VISION Cloud software stack [27]. The VISISON Cloud project focused on efficiently storing large object, which are available in many domains: analysis images in the medical domain or Virtual Machine images in the Cloud computing area. VISION Cloud has been chosen as a basis because it offers an interesting and appealing metadata concept as an integral part for handling storage entities: Metadata can be attached to containers (including large objects and other containers) and objects to enable an efficient retrieval in first place. However, the metadata concept is also useful for controlling data placement. The key idea is to attach storage requirements as metadata as well to containers and objects that an appropriate storage solution should satisfy. The storage requirements are specified at a higher semantic level instead of more technical physical properties.

The overall approach has been described in detail focusing on what VISION Cloud concept can be reused and what additional component is required. In general, VISION Cloud and its storage system architecture are well-suited to support a multi-criteria approach. The major component of mapping user requirements to more physical storage parameters can easily be integrated into the VISION Cloud architecture.

Our future work will focus on evaluating the effectiveness of the multi-criteria approach and its included monitoring and recommendation system. In particular, we want to investigate in more detail how effective users can be shielded from complex price schemes of public Cloud providers, i.e., whether reasonable mappings from price aspects to a cost user criterion can be found. Moreover, we think of extending the approach to a self-learning system that uses information about the user's satisfaction to adopt the mapping table between users' requirements and the properties of the storage system.

Acknowledgements. The research leading to the results presented in this paper has partially received funding from the European Union's Seventh Framework Programme (FP7 2007-2013) Project VISION Cloud under grant agreement number 217019 and is accordingly based on its deliverables and research publications as citations have pointed out.

References

1. Abu-Libdeh, H., Princehouse, L., Weatherspoon, H.: RACS: a case for cloud storage diversity. In: Proceedings of the 1st ACM Symposium on Cloud Computing, SoCC 2010, pp. 229–240. ACM, New York (2010)
2. Apache Libcloud: a unified interface to the cloud. http://libcloud.apache.org/. Accessed 15 Nov 2017
3. Bermbach, D., Klems, M., Tai, S., Menzel, M.: Metastorage: a federated cloud storage system to manage consistency-latency tradeoffs. In: Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing, CLOUD 2011, pp. 452–459. IEEE Computer Society, Washington, DC (2011)
4. Brantner, M., Florescu, D., Graf, D., Kossmann, D., Kraska, T.: Building a database on S3. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, p. 251 (2008)
5. Broberg, J., Buyya, R., Tari, Z.: Creating a 'Cloud Storage' mashup for high performance, low cost content delivery. In: Feuerlicht, G., Lamersdorf, W. (eds.) ICSOC 2008. LNCS, vol. 5472, pp. 178–183. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01247-1_17
6. Bunch, C., et al.: An evaluation of distributed datastores using the appscale cloud platform. In: Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing, CLOUD 2010, pp. 305–312. IEEE Computer Society, Washington, DC (2010)
7. CDMI: Cloud data management interface version 1.1.1. https://www.snia.org/sites/default/files/CDMI_Spec_v1.1.1.pdf. Accessed 15 Nov 2017
8. CloudSwitch Homepage. http://www.cloudswitch.com. Accessed 15 Nov 2017
9. Deltacloud Homepage. http://deltacloud.apache.org/. Accessed 15 Nov 2017
10. Fielding, R., Taylor, R.: Principled design of the modern web architecture. ACM Trans. Internet Technol. **2**(2), 115–150 (2002)
11. Fox, A., et al.: Above the clouds: a Berkeley view of cloud computing. Report UCB/EECS, 28, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley (2009)
12. Gogouvitis, S.V., Katsaros, G., Kyriazis, D., Voulodimos, A., Talyansky, R., Varvarigou, T.: Retrieving, storing, correlating and distributing information for cloud management. In: Vanmechelen, K., Altmann, J., Rana, O.F. (eds.) GECON 2012. LNCS, vol. 7714, pp. 114–124. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35194-5_9
13. Hohenstein, U., Jaeger, M., Dippl, S., Bahar, E., Vernik, G., Kolodner, E.: An approach for hybrid clouds using vision cloud federation. In: 5th International Conference on Cloud Computing, GRIDs, and Virtualization (Cloud Computing), Venice, pp. 100–107 (2014)
14. Hohenstein, U., Jaeger, M., Gogouvitis, S.: A multi-criteria approach for large-object cloud storage. In: 6th International Conference on Data Science, Technology and Applications (DATA 2017), Madrid, Spain, pp. 75–86 (2017)
15. Jaeger, M.C., Messina, A., Lorenz, M., Gogouvitis, S.V., Kyriazis, D., Kolodner, E.K., Suk, X., Bahar, E.: Cloud-based content centric storage for large systems. In: Proceedings of Federated Conference on Computer Science and Information Systems - FedCSIS 2012, Wroclaw, Poland, 9–12 September 2012, pp. 987–994 (2012)
16. Kolodner, E., et al.: A cloud environment for data intensive storage services. In: CloudCom, pp. 357–366 (2011)
17. Kolodner, E., et al.: Data intensive storage services on clouds: limitations, challenges and enablers. In: Petcu, D., Vazquez-Poletti, J. (eds.) European Research Activities in Cloud Computing, pp. 68–96. Cambridge Scholars Publishing, New Castle upon Tyne (2011)

18. Mell, P., Grance, T.: The NIST definition of cloud computing. NIST special publication 800-145, Sept 2011
19. Nasuni Homepage. http://www.nasuni.com/. Accessed 15 Nov 2017
20. Nimbula Homepage. http://en.wikipedia.org/wiki/Nimbula. Accessed 15 Nov 2017
21. Nirvanix Homepage. http://www.nirvanix.com/products-services/cloudcomplete-hybrid-cloud-storage/index.aspx. Accessed 15 Nov 2017
22. NoSQL databases. http://nosql-database.org. Accessed 15 Nov 2017
23. Sandalage, P., Fowler, M.: NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Pearson Education, Upper Saddle River (2013)
24. Sheth, A., Larson, J.: Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Comput. Surv. **22**(3), 183–236 (1990)
25. SmeStorage Homepage. https://code.google.com/p/smestorage. Accessed 15 Nov 2017
26. Vernik, G., et al.: Data on-boarding in federated storage clouds. In: Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing, CLOUD 2013, pp. 244–251. IEEE Computer Society, Washington, DC (2013)
27. VISION-Cloud project consortium: high level architectural specification release 1.0, vision cloud project deliverable D10.2, June 2011. http://www.visioncloud.com. Accessed 15 May 2017
28. VISION-Cloud project consortium: data access layer: design and open specification release 2.0, deliverable D30.3b, Sept 2012. http://www.visioncloud.com/. Accessed 15 May 2017
29. Junker, U.: QUICKXPLAIN: preferred explanations and relaxations for over-constrained problems. In: Proceedings of the 19th National Conference on Artificial Intelligence (AAAI 2004), pp. 167–172 (2004)
30. ZeroMQ. http://zeromq.org/. Accessed 15 Nov 2017

Player Performance Evaluation in Team-Based First-Person Shooter eSport

David Bednárek, Martin Kruliš, Jakub Yaghob, and Filip Zavoral(✉)

Parallel Architectures/Algorithms/Applications Research Group,
Charles University, Prague, Czech Republic
{bednarek,krulis,yaghob,zavoral}@ksi.mff.cuni.cz,
http://www.ksi.mff.cuni.cz/parg

Abstract. Electronic sports or pro gaming have become very popular in this millenium and the increased value of this new industry is attracting investors with various interests. One of these interest is game betting, which requires player and team rating, game result predictions, and fraud detection techniques. This paper discusses several aspects of analysis of game recordings in Counter-Strike: Global Offensive game including decoding the game recordings, matching of different sources of player data, quantifying player performance, and evaluation of economical aspects of the game.

Keywords: E-sport · Data analysis · Data integration · Data quality
Player rating

1 Introduction

Competition between people has been one of the defining characteristics of the entire human race. In the digital era, one of the domains where people express their competition is computer gaming. In fact, game competitions have become much more than common leisure activities and professional players earn money for attending tournaments similarly to professional athletes. A new industry was founded around game competition which is commonly refered to as *electronic sports* (eSports) or *pro gaming*.

Similarly to traditional sports, additional activities have joined eSports domain, such as fan clubs, product advertisement (i.e., propagation of products/companies at game tournaments), or betting on game results. These activities depend on additional support from IT industry for data processing and analysis, especially providing publishing infrastructure (broadcasting the eSport events), player rating, game result prediction, and fraud detection.

In our work, we focus on analysing data from played games in order to rate the performance of individual players and teams. These data can be subsequently used for player rating, team rating, preciting game results, or even for fraud detection. This paper addresses various aspects of processing the data of *Counter-Strike: Global Offensive* (CS:GO) recorded games and to match these data with existing sources such as HLTV[1] player boards.

This paper extends our work in [1]; it is organized as follows. Section 2 explains the rules and technical details of the CS:GO game and reviews existing datasets and matching methods. Details of data parsing and preprocessing are summarized in Sects. 3, and 4 describes our player-matching technique which provides integration with HLTV data source. Computing of players' rating and its verification is described in Sects. 5 and 6 analyzes the impact of the in-game economy to the game results. Section 7 summarizes our findings and concludes the paper.

2 Counter Strike: Global Offensive

The *Counter Strike: Global Offensive* (CS:GO) is a first-person shooter game where two teams of 5 players compete. The game has several scenarios, but the only one used in eSport is called *bomb defusal.* One team plays the *terrorists* who are attempting to plant a bomb at one of two possible targets (*planting sites*). The other team plays *counter-terrorists* who are trying to prevent the terrorists from planting and detonating the bomb.

The game itself aims to be rather realistic. The arsenal of available weapons and equipment is based on real weapons and equipment and the game attempts to simulate their actual behavior. The players are limited in the amount of things they can carry in the same way as real humans would be. Finally, when a player is killed in the game, he/she is dead till the end of the round (i.e., there is no quick respawning of players).

The game is played on several[2] well known maps. Based on the characteristics of individual tournaments, the teams select one or several maps (typically 3) by a deterministic negotiation protocol. The game is played once for each map and a winner of each map is recognized. The team that wins certain amount of maps (e.g., 2 wins on 3 maps) wins the whole match. In the remaining text, we will use the term '*game*' instead of '*match*' since word '*match*' may get somewhat ambiguous when describing player matching.

Multiple rounds are played on each map. Each round ends with a victory of one side and the winning team is awarded one point. Terrorists win if they detonate the bomb or kill all counter-terrorists. Counter-terrorists win if they kill all terrorists or prevent them from detonating the bomb within given time limit (about two minutes). The game ends when a team gets 16 winning points and the score of the teams differs by at least two. If score 16 : 15 is reached, an overtime is issued using rather complex rules. There are also rules for switching sides, so both teams play several rounds as terrorists and several rounds as counter-terrorists. Each team may *buy* weapons and equipment for virtual money before each round. The virtual money are earned in the game for winning rounds and

[1] www.hltv.org.

[2] In most tournaments, the map pool is limited to 7 standard maps.

killing opponents. Furthermore, players who survive the round keep most of their equipment. In Sect. 6, we try to quantify the impact of the in-game economy on the outcome of the game.

2.1 Demofiles and Data Sources

A game (one map) can be recorded into a DEM file called *demo file*. It is basically a serialization of the data transferred over the network between the server and the players. The demofile can be recorded by the server itself, but it can also be recorded by a spectator (player present in the game which is invisible to other players and cannot affect the game).

The demofile for the analysis must be provided by the tournament organizer, or a spectator access must be granted to the game server. The demofile can be also processed on the fly when the game is running (i.e., read whilst it is being written) to provide real-time game analysis. A very similar data stream is provided by *GOTV* – a broadcasting channel integrated in the game, which may be enabled on the server and it broadcasts game data to subscribed spectators. However, many tournaments delay this data (e.g., by 90 s), so the data cannot be fed back to the players via covert channels such as phone.

There is also a huge community interested in CS:GO which manage data about players and games. Perhaps the largest site dedicated to this game is HLTV. It registers all important events and tournaments and gather results. The site also gathers recorded demofiles and provide them for download. Unfortunately, HLTV administrators have little interest in sharing the data on a large scale; hence, there is no API and all data has to be scraped from web pages. Another issue with HLTV data is the player matching – i.e., interlinking existing player profiles with players in demofiles. Despite the fact that the site has identified the players internally, there is no direct linkage between the demofiles and the web.

HLTV player matching is a special case of a more general data matching problem. Although many commercial, open-source, or research data-matching systems have been developed, such as BigMatch, D-Dupe, R RecordLinkage, and many others [3], none of them is able to take into account the particular needs of HLTV matching. The problem is similar to nickname identification which is addressed especially in the domain of social networks. Some of the proposed methods use supervised learning methods [11], but they cannot be used in HLTV matching due to absence of the relevant labeled training data of sufficient size.

Other class of methods use their own specific models for matching individual accounts in particular social networks. These methods compute a similarity score from profile informations [7] or combine various identity search methods exploiting distinct profile attributes to match accounts across social networks [6]. All of these methods utilize additional information available in user profiles to match the accounts. To our best knowledge, none of the published methods could be applicable to HLTV matching as additional information are not available in demofiles. Therefore, we propose our own method which is described in Sect. 4.

3 Game Data Parsing

The game recordings are saved in *demo files* – a proprietary format of Valve Corp which basically capture all network traffic [2] between the game server and clients. A demo file is fixed to one map, so if multiple maps are played in a game, multiple demo files are required. On the other hand, it captures a period of time in a game, so the game on one map may be (and sometimes is) divided into multiple demo files. Demo file uses three levels of encoding: network packets, messages encoded using Google's Protocol Buffers [12], and a proprietary Valve's data compression.

All encoding levels are bitwise-oriented. Parsing one demo file must be done sequentially and only maintaining the decoding state itself is rather complicated. Furthermore, the third layer of encoding is not very well documented (as it is proprietary) and changes with new versions of the game.

The CS:GO server (called Source) uses a tick time unit as a logical time for the game simulation. All client inputs, actions, and interactions with each other and world objects are resolved periodically in these ticks. Typical tick-rate for tournament servers is 128 ticks per second (one tick lasts approximately 7.8 ms).

Probably due to size reasons, the demo file stores information from 8 subsequent ticks together in a single *burst*. This burst has two fixed parts – list of *events*, and list of *delta changes*. Both parts are quite important, so we describe them in more detail.

3.1 Events and Delta Changes

Events register important player actions and interactions with the simulated world – for instance, when player fires a weapon, bomb is planted, or the round concludes. Events are structures that carry all event-related data, such as location of the event, player who caused the event etc. These structures are typically simple to parse, thus events are logical choice for basic data analysis of the game. On the other hand, we have discovered that events are not completely reliable and they differ significantly across the game versions. Some important events (e.g., player deaths) are sometimes missing or contain invalid information.

The second part of each burst are the delta changes. Each entity in the game is represented by its own structure (e.g., a player has a structure which contains coordinates on the map, pitch, health, etc.). Delta changes basically forms an update transaction of these structures – i.e., a list of game objects and their properties which should be inserted, updated, or removed from the game.

Delta changes are much more complex to process as they do not carry a complete information, but only a change from the previous state. Therefore, to process the state of the game completely, we have developed a simplified game simulator which holds all game objects and apply the delta change lists on them. This way we can determine the complete state of the game at any time.

3.2 Data Quality

There are several important issues we have observed when processing demo files. Perhaps the most important issue is the fact that the demo files may be corrupted. Unfortunately, the lack of documentation prevent us to detect whether this is actually a problem of file corruption on the data level, a matter of protocol errors or old protocol constructions, or simply an insufficient knowledge of the format. In order to deal with this problem, we have defined a *trustworthiness* measure which is assigned to each parsed demo file. It is basically a maximum of parsing error severity levels from the file processing. Based on trustworthiness, the files were divided into three categories: files without errors, files which were parsed correctly but contain unexpected data values (e.g., players without identifiers), and files which cannot be parsed correctly. The first category can be fully processed while the last one cannot be processed at all. The files with unexpected values can be still partially processed for some statistical purposes (e.g., when computing global precision of weapon fire).

Even if the file can be parsed correctly, the fact that the data are aggregated into bursts which represent 8 subsequent ticks may still cause minor processing issue. For instance, if a player is spawned in tick T_1 and immediately shoots in a tick T_2 whilst both ticks T_1 and T_2 are in the same burst, it might happen that the information about weapon fire would preceded the information about player appearance. Sequential parser would fail in such case as it would encounter a weapon fire caused by nonexisting player. Fortunately, this seeming violation of causality can be easily rectified by deferring processing of events and delta changes that involve nonexisting objects after the entire burst is processed.

The whole demo file contains all data from the period of time when the recording was enabled (similarly to a camera). Typically, such recording is initiated well before the actual game is started. Therefore, it also contains warmup rounds and sessions when the players are connecting and waiting. These parts are often very disruptive for data processing as they contain similar events and delta changes as a regular game, but they should not be used for data analysis of the players' performance. Furthermore, in these time periods, the players typically connect and disconnect which makes it more difficult to determine, which players are actually participating in the game.

Fortunately, a reset is typically performed just before the actual game begins. In some cases, the game is reset multiple times in one demo file. It is imperative to found the last reset just before the game starts and then process only the data after this reset.

Finally, we have mentioned that the game events are not reliable, especially when dealing with older versions of demo files. On the other hand, delta changes are more tedious for processing. In order to maintain the simplicity of events but provide better reliability, we have place detectors on certain game object properties, and when these properties are changed, we generate our own events. Our simulated events mimic the structures of the parsed game events to simplify their subsequent processing.

Probably due to size reasons, the DEM format stores all informations from 8 server ticks together in a one burst. It brings the first major problem with data: we are not able to decide exact sequence of events and informations in the burst of 8 ticks. For example, two players are shooting each other. We cannot decide, who shoot first, if both events occur in the one burst.

Grouping 8 ticks together in one burst brings another major problem: dealing with not yet defined entities. The order of different kind of informations in the burst is hard-coded, i.e. the first set of information in the burst are always events, the second set are player informations, etc. Let us have a small example: a player spawns in a tick T_1 and immediately shoots in a tick T_2. Shooting is an event, events are always the first set of information in a burst. Player spawn is encoded in player informations, which is always the second set of information in the burst. Each player has assigned a unique number/identification at the player spawn. Events about players use player identification for encoding involved players. When both ticks T_1 and T_2 are in the same burst, we will encounter an event using a player identification, but we don't have a player with this identification yet. Therefore we have to deffer resolution of missing information in data for the whole burst.

This brings us to the third major problem: sometimes the information is just missing. It looks like the information stored in DEM file is not totally reliable and the stored information is not complete. We cope with this problem by computing *trustworthiness* index for each DEM file during the import of DEM files to the database. Its computation is based on detecting missing data and their importance for following decoding.

The last major problem is a protocol/encoding evolution with different versions of CS:GO. E.g. older versions don't send several important events like "player hit", etc., while newer versions of CS:GO send them. As those events are important for our player ranking computation, we have to retrieve the information somehow else. Fortunately, there is an other form of data stored in the DEM file even with the oldest versions, which contains all necessary informations required for the whole game simulation. Unfortunately, this form of data is very complicated, vast, and totally undocumented.

4 HLTV Integration and Player Matching

Our research included an interesting case of data integration – our data originated from the following sources:

The database of Steam users, maintained by the Valve Corporation. Although there is a public API for this database, many players have set their profiles to private and, therefore, we did not use this data directly. Nevertheless, the database was manifested in our data indirectly, since the game recordings contained Steam user identifiers for all players and observers. Since the Steam system associates the user identifiers with valuable assets like credit cards and purchased software, it is reasonable to assume that the mapping of Steam user identifiers to physical persons is sufficiently reliable.

The game recordings (demo files), created by the organizers of the tournaments. The demo files are created by the game server or by an on-site spectator client; although they are not protected by cryptographic means, their alteration would require considerable effort – thus, their contents may be considered sufficiently reliable. However, there is no 1:1 correspondence between demo files and games; therefore, some games may be covered incompletely and some demo files may contain more recordings than the game itself (e.g., a warmup phase).

The HLTV database, created by the community of tournament organizers, players, and fans. Due to the community origin, the reliability of data is variable, depending on the author of particular record and the amount of effort invested, leaving aside the possibility of intentionally entered false data. Links to the demo files are part of the HLTV database; therefore, the relation between demo files and games is unreliable as well, containing frequent omissions and a number of mismatched entries.

In the dataset extracted from the HLTV website in November 2016, there were 18 513 game entries; however, usable game recordings were available for only 8474 of them. There were 6566 HLTV player entries, but only 4888 of them were attached to at least one usable game entry. The 8474 recordings contained 6563 different Steam user identifiers.

The main problem in our case of data integration was matching player entries in game recordings to player entries in the HLTV database. For each game, there is a set of Steam users present in the recording and a set of HLTV player references entered by the community. The rules of game do not recognize particular roles of the players in the game, except for team allegiance. Consequently, there is no reliable mechanism which could pair the Steam users involved in a recording to the player entries present in the HLTV database for the same game.

4.1 Names and Nicks

When Steam users connect to the game server, they may select a name which is displayed in the messages and statistics produced by the game server. Users often choose fancy names to attract attention. There are some loosely followed conventions like including the team name in brackets; however, most of the names are created as a form of free art which may include a play on characters, sounds, or meanings of the name. On the other hand, there are names chosen in haste or neglect like 'asdf' or similar semi-random texts. In many cases, the same user uses different names in different games.

In the HLTV database, the visible identifier of a player is called a *nick*. Unlike in-game names, most nicks consist only of plain alphabetical words, i.e. no decorations are present. Nicks are rarely changed (and there is no record of previous nicks in the database).

Consequently, matching the Steam users to HLTV nicks requires a string-matching algorithm capable to compare the fancy in-game name to the plain nick; the system must also cope with the fact that in-game names change while the nicks are stable. It is also apparent that the string matching could not be the sole mechanism for player matching, since it is possible that in-game name would bear no similarity to the nick of the same physical person.

The name-nick matching requires a similarity measure – among the possible candidates for such a measure, the length of *Longest Common Subsequence* [5] was selected as a trade-off between quality and implementation complexity. The LCS measure is based solely on matching characters (unlike, for instance, the Levenstein distance), i.e. it does not penalize the presence of additional non-matching characters which frequently occur in fancy in-game names.

4.2 Formal Model

The available input data may be formalized as a tri-partite *game-presence graph*

$$G = (V_u, V_h, V_m, E_{um}, E_{hm})$$

where V_u, V_h, and V_m are sets of vertices corresponding to Steam users, HLTV player entries, and games, respectively. $E_{um} \subseteq V_u \times V_m$ are edges representing the presence of a Steam user in a game recording, $E_{hm} \subseteq V_h \times V_m$ are edges representing the presence of an HLTV player in a game.

Names and nicks are formalized as mappings $n_u : E_{um} \to \mathcal{S}$ and $n_h : V_h \to \mathcal{S}$ where $\mathcal{S}$ denotes the domain of strings. The in-game names are associated to edges because players may select a different name in each game while the HLTV nicks are bound to vertices representing the player entries. The LCS measure is denoted as $s_{LCS} : \mathcal{S} \times \mathcal{S} \to \mathcal{N}$.

The player matching process starts with determining *candidate pairs*:

$$C = \{\langle u, h\rangle \ ; \ (\exists m \in V_m)\langle u, m\rangle \in E_{um} \wedge \langle h, m\rangle \in E_{hm}\}$$

A pair of a Steam user and a HLTV player becomes a candidate if and only if there is a game where both participated. Each candidate pair receives a *weight* corresponding to the number of such games:

$$w(u, h) = |\{m \in V_m \ ; \ \langle u, m\rangle \in E_{um} \wedge \langle h, m\rangle \in E_{hm}\}|$$

Then, for each candidate pair, the LCS measure is computed:

$$s(u, h) = max\{s_{LCS}(n_u(u, m), n_h(h)) \ ; \ \langle u, m\rangle \in E_{um}\}$$

Since a Steam user u may use more than one name $n_u(u, m)$, all the names are compared with a HLTV player nick $n_h(h)$ and the maximum similarity is considered. Note that names which u have used in all its games are considered, including those games where the HLTV player h was not present – this arrangement improves the pairing process for users (incorrectly) represented by more than one HLTV entry.

4.3 Player-Matching Algorithm

The player matching algorithm makes use of both the game-presence graph (transformed into the weights $w(u,h)$) and the name-nick similarity $s(u,h)$. The weights are given priority, i.e. the name-nick similarity is used only to break ties in the weights.

Using the game-presence graph, subsets C_u, C_h, and C_b of the candidate-pair set C are named *u-best pairs*, *h-best pairs*, and *best pairs*, respectively, and defined by the following criteria:

$$\langle u,h\rangle \in C_u \;\Leftrightarrow\; \langle u,h\rangle \in C \;\wedge\; (\forall \langle u,h'\rangle \in C)\; (h' \neq h \Rightarrow (w(u,h') < w(u,h) \;\vee\; w(u,h') = w(u,h) \;\wedge\; s(u,h') < s(u,h)))$$

$$\langle u,h\rangle \in C_h \;\Leftrightarrow\; \langle u,h\rangle \in C \;\wedge\; (\forall \langle u',h\rangle \in C)\; (u' \neq u \Rightarrow (w(u',h) < w(u,h) \;\vee\; w(u',h) = w(u,h) \;\wedge\; s(u',h) < s(u,h)))$$

$$C_b = C_u \cap C_h$$

In other words, the candidate-pair set C is ordered by the lexicographical ordering on their $\langle w(u,h), s(u,h)\rangle$ values. Then, $\langle u,h\rangle$ is a u-best pair if all other $\langle u,h'\rangle$ pairs are positioned lower in the lexicographical ordering. Similarly, it is a h-best pair if all other $\langle u',h\rangle$ pairs are positioned strictly lower.

Our problem is similar to maximum weighted bipartite matching [8]; however, we are interested only in matchings which are strictly better than any other. If there is a tie among several maximum matchings, our problem requires that the uncertain part of the matching be removed, i.e. not paired at all. The Hungarian algorithm for maximum weighted bipartite matching cannot reliably detect the existence of alternative matchings during its score-improving phases; therefore, it is unusable in our settings.

Note that C_u is a partial mapping from V_u to V_h and C_h^{-1} is a partial mapping from V_h to V_u. If $C_h^{-1}(C_u(u)) = u$ then $\langle u, C_u(u)\rangle$ is considered a best pair. Note that the definition of C_u and C_h implies that the values of $\langle w(u,h), s(u,h)\rangle$ must be strictly increasing along any directed acyclic path formed by C_u and C_h^{-1} edges; therefore no cycle longer than two may exist and the best pairs are the only cycles in the oriented graph formed by C_u and C_h^{-1}.

The C_b relation does not necessarily cover all vertices in V_u or V_h, leaving a *residual pair set* C_r defined as

$$\langle u,h\rangle \in C_r \;\Leftrightarrow\; (\forall h')\; \langle u,h'\rangle \notin C_b \;\wedge\; (\forall u')\; \langle u',h\rangle \notin C_b$$

The residual pair set is then used as a new candidate-pair set input for the search for best pairs and the process is repeated until no new best pairs are found. This produces a sequence of candidate-pair sets $\{C^{(i)}\}$ defined by

$$C^{(1)} = C \qquad C^{(i+1)} = C_r^{(i)}$$

and terminated when $C_b^{(i)} = \emptyset$.

The rationale behind the iteration is the informal definition "a pairing is best if all better candidates were already paired". Such a definition is recursive with negation – the absence of rigorous semantics for such a definition is solved by the stratification introduced by the iterative algorithm.

The sequence of best-pair sets $\{C_b^{(i)}\}$ produces the final *pairing* (by disjoint union)

$$P = \bigcup \{C_b^{(i)}\}$$

together with the following *confidence attributes* assigned to each $\langle u, h\rangle \in C_b^{(i)}$:

$$a(u,h) = \langle w(u,h), s(u,h), i\rangle$$

4.4 Evaluation

The confidence attributes w, s, and i are used to estimate the matching quality of the pair, using the following empirical formula:

$$q(u,h) = \frac{0.25 \cdot w(u,h)}{deg_G(u)} + \frac{0.25 \cdot s(u,h)}{|n_h(h)|} + \frac{0.5}{i(u,h)}$$

The first term of the formula expresses the ratio of games where both u and h participated to all games of the user u (i.e. the degree of vertex u in the game-presence graph G). The asymmetry towards u (i.e. the ignorance of $deg_G(h)$) corresponds to the fact that the HLTV database was designed to have only one entry for a physical person while the same person may have more than one Steam user account. Thus, the ratio becomes 1 if the HLTV player h is present in all games played by the Steam user u.

The second term compares the length of the longest common subsequence of $n_u(h,m)$ and $n_h(h)$ to the length of $n_h(h)$. The most perfect matching occurs when $n_h(h)$ is contained in $n_u(h,m)$ (for some game m) – in this case, the ratio equals 1.

At the input of our player matching algorithm, there were 6563 Steam users and 4888 HLTV player entries which together participated in 8474 recorded games. We worked with the idea that each HLTV player corresponds to one Steam user, i.e. we searched only for 1:1 matchings.

Our algorithm was able to find 4775 pairs – it means that 97.69% of HLTV player entries were matched to Steam users. Furthermore, the success ratio improves to 99.70% when frequent players with 10 or more games are considered.

5 Player Rating

One of the main goals in this project was a proposal of players performance classification in particular matches. The resulting formula should satisfy expectations, such as high ranking of the best players, or higher compound ranking of the winning team. The classification could be then used for result predictions, suspected fraud detection, etc.

5.1 HLTV Rating

HLTV rating [9] is used for measuring and publishing long-term performance of players and their comparison. It disregards the amount of rounds played in a match, as it considers average values. It has a limited range, as it can go from 0 to 3 (although rarely over 2, which is then an amazing performance). It also has a well spread range of values that should reflect properly on how many average, good and great performances there are.

HLTV rating formula (RWMK stands for RoundsWithMultipleKills) is as:

$$Rating = \frac{KillRating + 0.7 * SurvivalRating + RWMK}{2.7}$$

where

$$KillRating = \frac{Kills/Rounds}{0.679 * AverageKillsPerRound}$$

$$SurvivalRating = \frac{(Rounds - Deaths)/Rounds}{0.317 * AvgSurvivedRoundsPerRound}$$

$$RWMK = \frac{\sum_{i=1}^{5}(i^2 * K_i)/Rounds}{1.277 * avg\left(\sum_{i=1}^{5}(i^2 * K_i)/Rounds\right)}$$

Ki = number of rounds with i kills

Although this formula means substantial simplification of the player performance and usefulness, in most cases the best players gain the highest HLTV rating. Nevertheless, according to HLTV, in some cases the HLTV rating underestimates the real player's value, especially for players in special roles, such as Entry Fragger, Refragger/Support or Strategy Caller. Therefore, in June 2017, HLTV introduced the new Rating 2.0 [10] that is based on more activities leading to the team success. Namely, partial ratings based on average damage per round and percentage of rounds with a kill, assist, survival or traded death were introduced. Unfortunately, the exact formula is not (yet) published.

5.2 Individual Ratings Rer Round

In order to compute individual ratings in one game, we consecutively used several methods to compute individual round, game and match ratings.

First, the HLTV rating formula was computed for each player in each round/game/match played. The HLTV formula is intended for long-term players' evaluation; therefore it is not very suitable for local rating. Nevertheless, the cumulative individual HLTV ratings were correlated to the team results surprisingly well.

One of the important events having a significant impact on the round result is the first kill; loosing one player usually skews the probability in favor of the killing team. For our ratings, we add a special bonus for the first kill. Although this is very simple part of the overall rating, it can improve the validity of the rating. See Sect. 5.7 for validity evaluation.

On the other hand, some other explored methods did not contribute to the final rating. We computed several statistics of accuracy and efficiency of shooting (even segmented by the type of weapons) did not result into considerable correlation with team results. Similarly, no combination of weights of refrags (a player was shot down immediately after he kills an opponent player) was beneficial.

5.3 On HLTV Rating Improvements

Beside consecutive improvements of the original HLTV rating, we explored a new approach more tightly coupled to particular game situations and their importance. Our improvement is based on an observation that player kills may have different meaning (in the sense of game strategy) based on their location on the map. For instance, a kill that occurs in an alley during one-to-one encounter may not affect the game directly, whilst a kill that occurs on a strategically important spot may prove to be the turning point of the game. Hence, we made an attempt to statistically identify important locations and prioritize the value of kills (positively for the attacker, negatively for the victim) in the player rating.

5.4 Computing Cluster-Based Heatmaps

For the purposes of our improved rating, we compute *death heatmaps* – location information regarding where the players die the most and how the result of the game round in which the death occured. The heatmaps are computed individually for each game map and for each side (Terrorists and Counterterrorist). Since there are not enough data for a regular grid-based heatmaps, we have used clustering approach instead.

Each death event has 3D coordinates of the location where the player has deceased and his yaw and pitch values. The orientation is not that important in the case of death, so we have used the 3D coordinates only. A modified k-means algorithm [4] computes centroids of death clusters and each event is assigned to its nearest centroid. Since the k-means algorithm expects the number of clusters (k) as a parameters and we do not know the optimal number of clusters in advance we have used the following modifications:

- The parameter k represents the initial number of clusters and the final number may be lower.
- Parameter *minSize* is introduced. If the number of assigned events to a cluster is lesser or equal to *minSize*, the cluster is dismissed and the events are reassigned in the next iteration.
- Parameter *joinDist* is introduced. If two centroids have their euclidean distance lower than the *joinDist* threshold, their respective clusters are merged.
- Parameter *outlierDist* is introduced. If an event is further than *outlierDist* from its nearest centroid, it is marked as an outlier (i.e., it does not belong to any cluster).

– We also limit the maximal number of iterations in case a stable solution of the voronoi partitioning is not found quickly.

We have experimented with the parameters. The following combination was determined empirically with the insights of a domain expert and it has been proven applicable:

Parameter	Value
k	1000
iteration limit	10
minSize	5
joinDist	1.0
outlierDist	100

However, the parameter space was not searched exhaustively as the evaluation of each combination of parameters is rather time consuming.

Once the centroids and the assignments are computed, we compute values C_i and T_i for each cluster. The C_i is the number of death events in cluster i which were recorded in the rounds that were won by counterterrorist. Analogically, T_i holds the number of events from rounds which were won by the terrorists. Finally, we compute the *heat* value of the cluster as $h_i = (C_i - T_i)/(C_i + T_i)$ and the weight value (w_i), which is equal to the number of events assigned to the cluster.

5.5 Rating Death Events

To improve the player rating based on kills, we need to rate the death events in the game. The death event rating reflects the player rating of the killer positively and the rating of the victim negatively. We also consider friendly-fire kills, which affect the rating of the attacker in a negative way and do not affect the victim at all.

A death event computes its rating from the heat and weight values of the heatmap. The values are aggregated from all near-by clusters in the terms of euclidean distance. Closer clusters affect the event rating more than distant clusters. Hence, the death event rating e is computed as follows.

$$e = \frac{\sum_{i \in M'} \frac{h_i w_i}{d_i^2}}{\sum_{i \in M'} w_i}$$

The d_i is saturated Euclidean (L_2) distance between the location of the rated death and the centroid of cluster i. The saturation prevents underflow problems with float values by replacing all distances lesser than 1 to value 1. The K' stands for the subset of clusters[3] $K' \subseteq K$, centroids of which are within the threshold distance (*outlierDist*) from the death event location.

[3] Normally, we would use C to denote set of clusters, but we have selected K as they are produced by k-means to avoid ambiguity with values C_i computed for counterterrorists.

5.6 Player Rating from Death Events

The values from the heatmap have to be calculated in for player rating with different sings based on whether the killer is a **T**errorist or **C**ounterterrorist and based on whether the death was a frag or a friendly fire. Furthermore, we normalize the value from $\langle -1, 1\rangle$ to $\langle 0, 1\rangle$ interval linearly. The following table shows, how a death event rating e is contributes to the rating of the *attacker* and of the *victim* (we denote the contributing value $e_{attacker}$ and e_{victim} respectively).

attacker	victim	$e_{attacker}$	e_{victim}
C	T	$(e+1)/2$	$-(e+1)/2$
T	C	$(1-e)/2$	$-(1-e)/2$
C	C	$-(1-e)/2$	0
T	T	$-(e+1)/2$	0

The player rating r for given round is subsequently computed as

$$r = \sum_{frags} e_{attacker} \cdot M + \sum_{deaths} e_{victim} \cdot M,$$

where *frags* are death events where the player is the attacker, *deaths* are death events of that player (which is actually an empty set or a set containing exactly one event, but it is more convenient to write the formula using sums), and M stands for the *team membership factor* which is computed as the number of live players in the opponent team divided by the number of live players in the team of rated player just before the death event occurs.

The team membership factor rates the importance of the death event given the current team configuration. Similarly to the HLTV rating, it prioritizes later kills over early kills. However, this factor captures relative situation in both teams rather than simply counting deaths in a team. To explain the reasoning behind this, let us present an example. Team A has still 5 live members, but team B has only one survivor left. In such case, if player from A kills the last member of B, the frag is not that difficult to achieve (strictly when rating frags) as it is much easier to ambush a player who is outnumbered 5 : 1. On the other hand, if the last player of B kills one of A players, it may be considered a heroic act.

5.7 Emirical Evaluation

To evaluate proposed ratings, we have established the following methodology. The rating algorithms described above are defined to rate one player in one round. Thus we define player rating in a game as a simple arithmetic average of the rating from all rounds. Furthermore, we define team rating in a game as average rating of all players in a team.

In general, teams with better ratings should perform better than teams with lower ratings. Therefore, we should observe a regression between the difference in team rating and the final score. Let us denote Δr_G the difference between ratings of teams ($r_{teamA} - r_{teamB}$) in game G and $Score_G$ the final score (rounds won by team A minus rounds won by team B) of the game G. Figure 1 illustrates the idea by visualizing Δr and $Score$ values for all games in a 2D plot.

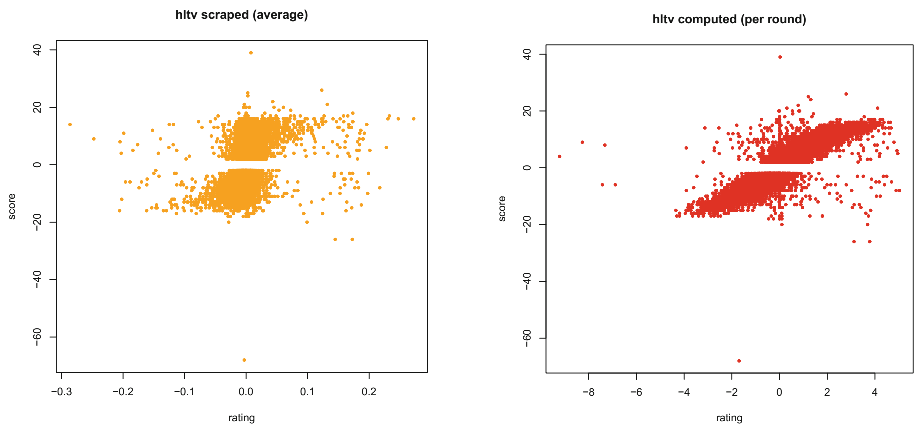


Fig. 1. Regression of HTLV rating and game score.

The left graph shows results when the HLTV rating values are taken from the official HLTV site. These values are less accurate, as they represent long term averages and do not reflect current games. The right graph visualize data where the HLTV ratings were computed separately for each round.

To compare individual rating methods, we have used simple linear regression method with least square residuals. The mean squared residual value is subsequently used as the norm that determines the quality of the result. Using this methodology, we have determined, that the proposed approach of clustering death events have similar quality as the original HLTV (computed per round). However, we have observed that both ratings produce often quite different results. Therefore, we have linear combinations of proposed methods.

Using empirical approach and the mean squared residual values as the fitness, the following combination was determined as the best linear combination.

$$r_{final} = 2r_{hltv} + r_{localized} + 3/2r_{first},$$

where r_{hltv} is the HLTV rating, $r_{localized}$ is our rating based on death event localization, and r_{first} is the first-kill rating (it equals to 1 if the player has made the first kill in the round and 0 otherwise). Figure 2 presents the $r_{localized}$ rating (on the left) and the final combined rating on the right.

5.8 Dead Ends and Possible Improvements

He have left more elaborate comparisons to our future work. However, we have tried several things that did not perform to initial expectations.

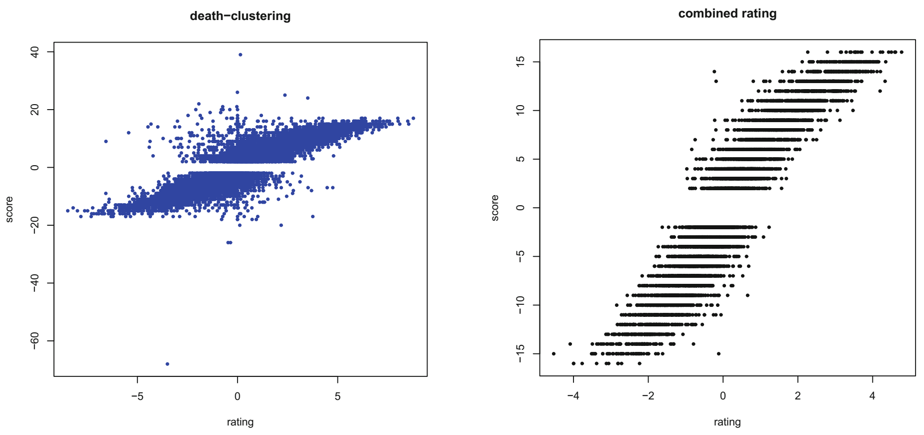


Fig. 2. Regression of the rating based on death event localization and the final combined rating.

First of all, we have tried adding some weight to the computed rating, which would indicate, how sure we are about the rating. We define *relevance* of each value v from the heatmap as

$$relevance = 1 - e^{-k\left(\frac{\sum_{i \in C'} h_i w_i}{\sum_{i \in C'} w_i}\right)^2}.$$

It is computed similarly like the heatmap value itself, but it does not account for the distance ($1/d^2$ is missing) and it projected to gaussian curve, so that the values near -1 and 1 are considered very relevant and the values near 0 (i.e., where the death events does not really say who will win) are not relevant.

The relevance was originally intended so that the $v_{attacker}$ will be multiplied by its relevance when the rating of a player is computed, but such operation actually decreased the quality of the result slightly.

The relevance is also used to compute the weight of a player's rating:

$$weight = \frac{\sum_{kills} v_{attacker} \cdot relevance + \sum_{death} v_{victim}}{|kills| + |death|}$$

The weight was originally intended to be incorporated into the final rating (r_{final}), for instance, if our rating is considered relevant, it will be used, otherwise HLTV rating will be used. However, we did not find a reliable threshold when to switch from one to another nor a weighted combination of both ratings (where weight is used for our rating and $1 - weight$ is used for HLTV) achieved better quality than the simple equation for r_{final} presented above.

6 Economy in the Game

Each CS:GO game consist of at least 16 rounds played on the same map; the teams switch sides every 15 rounds. Within each 15-round streak, the rounds are not independent: Each player has a sum of money and a set of weapons which propagate between rounds and change as a result of the game action. Money are awarded for the sole participation in each round, for some actions during each round, and for the victory in a round. The sole purpose of money is buying weapons between rounds. Unlike money, weapons may be transferred between members of the team, retrieved from dead opponents, or voluntarily abandoned. Weapons cannot be changed back for money and are lost when a player dies. Thus, the game has its economy side which constitutes an important part of team strategy and which must be considered when analyzing the performance of teams in individual rounds.

Because of the developing sets of weapons, each round is different and the probability distribution of the round outcome is also varied. An interesting question is therefore quantifying the influence of the weapon sets on the round outcome. However, the assignment of weapons to individual players forms a multi-dimensional space which is too big compared with the amount of data available (we have about 8000 rounds recorded per a map).

Therefore, we have chosen money as a proxy to the availability of weapons for a team. More precisely, we sum the amount of money in the pockets of team members and the price of weapons the team holds. We sample these quantities before the shopping phase, immediately after the preceding round – this corresponds to the fact that teams may adjust the weapon purchase according to the strategy for the following round as well as select strategy based on available money.

The total money is however an imperfect proxy, since this approach neglects the fact that money can not be transferred between players (which prohibits concentrating money for expensive purchases) and that weapons can not be monetized by selling back.

By using the total money proxy, we reduce the state space to two dimensions, M^T and M^C corresponding to the wealth of the two teams. Our input data consist of a sequence $\{D_1, \ldots, D_k\}$ of recorded rounds, where each round is described as a triplet $D_i = \langle M_i^T, M_i^C, R_i \rangle$ containing the total money of the two teams before the round and the result of the round, where $R_i = 1$ denotes victory of the T team and $R_i = 0$ the opposite.

Maximum wealth values observed in our recordings are about 100000 while the maximal price of a weapon is 5000. Consequently, we need to divide each axis into at least 20 intervals to make the statistics useful, i.e. dividing the state space into at least 400 buckets.

Our goal is determining the probability $p(M^T, M^C)$ that the upcoming round will be won by the T team. More precisely, we need a lower bound $p_L(M^T, M^C)$ and an upper bound $p_U(M^T, M^C)$ such that

$$p_L(M^T, M^C) \leq p(M^T, M^C) \leq p_U(M^T, M^C)$$

will hold with a given confidence level.

The easiest statistics would be considering each bucket independently, examining only the rounds played with $\langle M^T, M^C \rangle$ values within the selected bucket.

However, as the buckets are filled with rounds irregularly, many buckets receive only few rounds and their independent analysis would not be statistically significant. Therefore, more complex analysis is needed.

Our statistical analysis is based on the natural observation that having more money may only improve performance of a team. Consequently, our prior knowledge is the *monotonicity* of the probability $p(M^T, M^C)$, as expressed by the following implication:

$$M_1^T \leq M_2^T \wedge M_1^C \geq M_2^C \Rightarrow p(M_1^T, M_1^C) \leq p(M_2^T, M_2^C)$$

This monotonicity assumption allows us to estimate the lower bound $p_L(M^T, M^C)$ based on observations $\langle M_i^T, M_i^C, R_i \rangle$ whose indices are in the set

$$S_L(M^T, M^C) = \{ \ i \in \{1, \ldots, k\} \mid M_i^T \leq M^T \wedge M_i^C \geq M^C \ \}$$

Similarly, the upper bound $p_U(M^T, M^C)$ is computed from the set of observations

$$S_U(M^T, M^C) = \{ \ i \in \{1, \ldots, k\} \mid M_i^T \geq M^T \wedge M_i^C \leq M^C \ \}$$

Counting all the data satisfying the condition would generate bounds with high confidence levels; however, they would be too conservative because they would be based on data whose coordinates $\langle M_i^T, M_i^C \rangle$ may be significantly distant from the examined $\langle M^T, M^C \rangle$.

Therefore, our statistics is based on weighted influence of each data triplet (satisfying the condition above), using the weights

$$w_i = e^{-zd_i}$$

where

$$d_i = |M^T - M_i^T| + |M^C - M_i^C|$$

and $z = 0.00016$ is an *attenuation coefficient* selected so that the influence fades out to a half for each 4000 distance in team wealth. The value of z is based on the assumption that an additional weapon bought for 4000 may significantly change the outcome of a round. In statistical terms, the z value is considered a part of prior knowledge, not a result of statistical analysis.

The weighted average of relevant observed values is defined as

$$x_L(M^T, M^C) = \sum_{i \in S_L(M^T, M^C)} w_i R_i$$

$$x_U(M^T, M^C) = \sum_{i \in S_U(M^T, M^C)} w_i R_i$$

The R_i here are observed outcomes of individual rounds, i.e. random variables with Bernoulli distribution with the unknown parameter $p(M_i^T, M_i^C)$.

Because of the monotonicity (and the definition of the S_L set), the following inequality

$$p(M_i^T, M_i^C) \leq p(M^T, M^C)$$

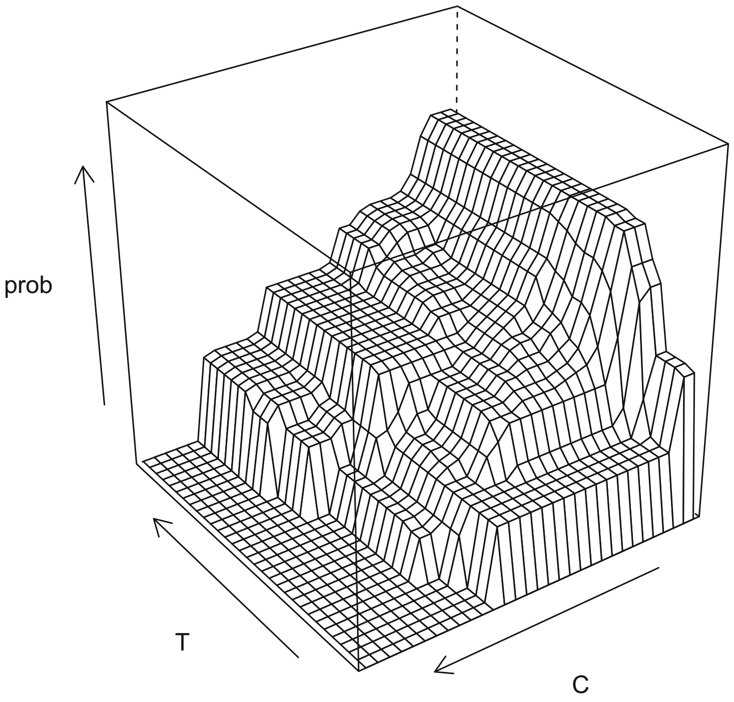


Fig. 3. Lower bound of T-win probability.

holds for all R_i which participate in the x_L sum. Therefore, the unknown cumulative distribution function (CDF) of x_L is certainly higher than the CDF of the following random variable

$$y_L(M^T, M^C) = \sum_{i \in S_L(M^T, M^C)} w_i Q_i$$

where all Q_i have the same Bernoulli distribution with $p(M^T, M^C)$.

This observation allows us to estimate the lower bound $p_L(M^T, M^C)$ as a function of the observed $x_L(M^T, M^C)$ by finding minimal $p(M^T, M^C)$ for which the probability that $y_L(M^T, M^C) <= x_L(M^T, M^C)$ is smaller than a chosen confidence parameter (0.95).

Since $y_L(M^T, M^C)$ is a weighted sum of variables with Bernoulli distributions, we can approximate it with a normal distribution whose parameters are dependent on the value of $p(M^T, M^C)$ and the set of w_i. This approximation leads to the following formula:

$$p_L(M^T, M^C) = \frac{2r_L + u_L - \sqrt{u_L(u_L + 4r_L(1 - r_L))}}{2 + 2u_L}$$

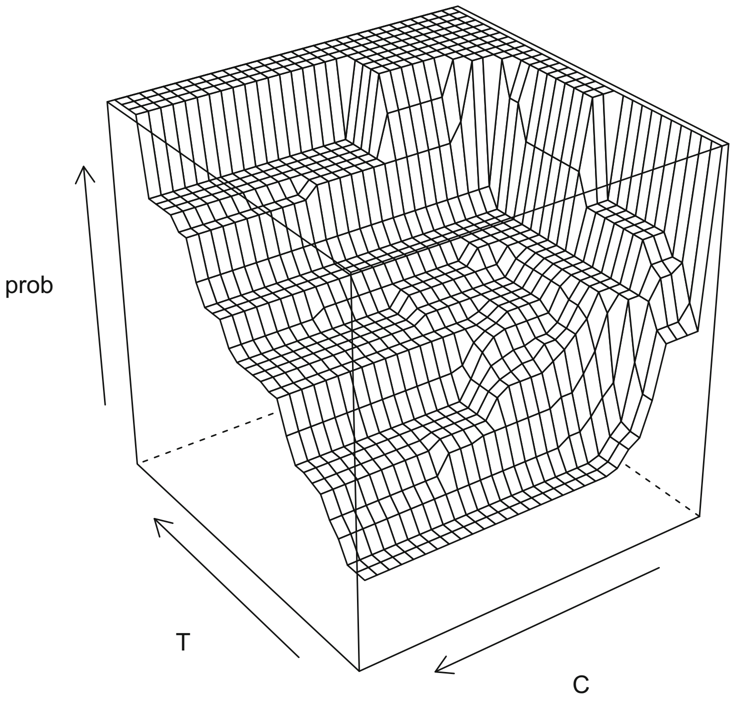


Fig. 4. Upper bound of T-win probability.

where

$$r_L = \frac{x_L(M^T, M^C)}{\sum_{i \in S_L(M^T, M^C)} w_i}$$

$$u_L = \alpha^2 \frac{\sum_{i \in S_L(M^T, M^C)} w_i^2}{(\sum_{i \in S_L(M^T, M^C)} w_i)^2}$$

where α is a parameter dependent on the required confidence level; the confidence level of 0.95 corresponds to $\alpha = 2.0$.

Similarly, for the upper bound:

$$p_U(M^T, M^C) = \frac{2r_U + u_U + \sqrt{u_U(u_U + 4r_U(1 - r_U))}}{2 + 2u_U}$$

where

$$r_U = \frac{x_U(M^T, M^C)}{\sum_{i \in S_U(M^T, M^C)} w_i}$$

$$u_U = \alpha^2 \frac{\sum_{i \in S_U(M^T, M^C)} w_i^2}{(\sum_{i \in S_U(M^T, M^C)} w_i)^2}$$

Figures 3 and 4 show the $p_L(M^T, M^C)$ and $p_U(M^T, M^C)$ values, respectively, computed from our data for a selected map (`de_mirage`). For best visibility, the 3D graphs are rotated so that the origin ($M^T = M^C = 0$) is shown on the right. The point ($M^T = 0$, $M^C = 120000$) is shown in the front, the point ($M^T = 120000$, $M^C = 0$), for which the probability of T-team victory is greatest, is shown in the back.

The graphs show several discontinuities parallel to either axis, which probably correspond to important wealth levels which allow the respective team to buy some advanced equipment. Exact determination of the equipment responsible for these discontinuities is a matter of future work.

7 Conclusions

In this paper, we have summarized several problems we encountered when we tried to analyze a set of game recordings and related data. Although the original goals of the research were player performance analysis and fraud detection, we found out that there are significant subproblems that must be solved before any performance analysis is attempted.

Besides technical aspects of the recordings, the subproblems include the fact that the player performance is dependent on the available weapons and, thus, on the in-game economy whose effect must be statistically quantified before any attempt to estimate the net performance of the players.

Furthermore, reliable player identification is needed for fraud detection as well as automated statistics keeping; therefore, integration of player data from multiple sources was necessary.

While we successfully solved the technical problems as well as player matching for the integration of data sources, the results in other areas are inconclusive. For the effect of in-game economy, our statistics yields only wide confidence intervals as illustrated in Figs. 3 and 4; for more precise estimation, larger datasets are probably necessary.

We have tested several approaches to the measurement of player performance based on their recorded activity. Advanced teams however use sophisticated strategies which often include sacrificing one of the players, whose performance therefore appears as poor in almost all statistic measurements available.

Therefore, for more accurate player performance analysis, understanding the team strategy is required. Our future work will focus on applying artificial intelligence methods in the team behavior and strategy analysis.

Acknowledgements. This work was supported by project PROGRES Q48.

References

1. Bednarek, D., Krulis, M., Yaghob, J., Zavoral, F.: Data preprocessing of eSport game records - counter-strike: Global offensive. In: Proceedings of the 6th International Conference on Data Science, Technology and Applications - Volume 1: DATA, pp. 269–276. INSTICC, SciTePress (2017)
2. Breu, L.: Online-games: Traffic analysis of popular game servers (counter strike: Source) (2007)
3. Christen, P.: Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31164-2
4. Hamerly, G., Elkan, C.: Alternatives to the k-means algorithm that find better clusterings. In: Proceedings of the Eleventh International Conference on Information and Knowledge Management, CIKM 2002, pp. 600–607. ACM, New York (2002). http://doi.acm.org/10.1145/584792.584890
5. Hirschberg, D.S.: A linear space algorithm for computing maximal common subsequences. Commun. ACM **18**(6), 341–343 (1975)
6. Jain, P., Kumaraguru, P., Joshi, A.: @i seek 'fb.me': Identifying users across multiple online social networks. In: Proceedings of the 22nd International Conference on World Wide Web, pp. 1259–1268. WWW 2013 Companion. ACM, New York (2013). http://doi.acm.org/10.1145/2487788.2488160
7. Jamjuntra, L., Chartsuwan, P., Wonglimsamut, P., Porkaew, K., Supasitthimethee, U.: Social network user identification. In: 2017 9th International Conference on Knowledge and Smart Technology (KST), pp. 132–137, February 2017
8. Kuhn, H.W.: The hungarian method for the assignment problem. Nav. Res. Logistics Q. **2**(1–2), 83–97 (1955). http://dx.doi.org/10.1002/nav.3800020109
9. Milovanovic, P.: What is that rating thing in stats? (2010). https://www.hltv.org/news/4094/what-is-that-rating-thing-in-stats. Accessed 21 Nov 2017
10. Milovanovic, P.: Introducing Rating 2.0 (2017). https://www.hltv.org/news/20695/introducing-rating-20. Accessed 21 Nov 2017
11. Peled, O., Fire, M., Rokach, L., Elovici, Y.: Entity matching in online social networks. In: 2013 International Conference on Social Computing, pp. 339–344, September 2013
12. Varda, K.: Protocol buffers: Googles data interchange format. Google Open Source Blog, Available at least as early as July 2008

Utilization Measures in a Learning Management System

Floriana Meluso, Paolo Avogadro(✉), Silvia Calegari, and Matteo Dominoni

Università degli Studi di Milano-Bicocca, 22104 Milan, Italy
paolo.avogadro@unimib.it, calegari@disco.unimib.it

Abstract. Learning Management Systems (LMSs) are becoming more and more popular and incorporate many different functionalities. For this reason, an evaluation of the quantitative utilization of all the parts of a LMS is essential. In this research we propose indicators and techniques which allow to understand in detail how a functionality is accessed by the users. These analytic tools are useful in particular for the administrators of the LMS which are in charge of allocating resources according to the workload and importance of the functionalities. We tested the proposed indicators with the data obtained from the LMS of Università degli Studi di Milano-Bicocca (Milan, Italy) about the messaging functionality. Although the students' messages can potentially be a source of big data, in the present case it is observed that the utilization is limited. With this analysis it has been possible to notice a similarity between the utilization of the message system and the empirical Zipf law. We also introduced the description of the structure of a dashboard which allows to access to the indicators and goes towards the definition of a global tool for students, teachers and administrators.

Keywords: Learning analytics · LMS · Indicator · Message system Zipf

1 Introduction

The term learning analytics (LA) includes many research fields, such as process mining, business intelligence, data processing, information retrieval, technology-enhanced learning, educational data mining and data visualization [13,14]. Indicators and visualization tools are commonly employed to understand, control and predict [11] the processes related to the learning activities for institutions at different academic levels ranging from primary schools, to universities and including all the learning cases, for example workplaces, etc. LA are expected to provide insights for students allowing them to take full control of their own learning, to give them a better idea of their current performance in real-time and to help them to make informed decisions about the study path [15,16].

LA are also very useful for teachers to determine, for example expert users, at-risk students, etc. [3,17]. The learning process is not static and changes with

J. Filipe et al. (Eds.): DATA 2017, CCIS 814, pp. 176–196, 2018.
https://doi.org/10.1007/978-3-319-94809-6_9

time, for this reason it requires a constant monitoring, evaluation, and adaption to the requests and needs of the stakeholders to guarantee high quality and ad-hoc outcomes [8]. Among the most important paradigms gathering popularity in education community there is the Flipped Learning (FL) [6], which is considered as an extension of the flipped classroom paradigm where a key role is assumed by the social features within the learning practice. In this context LA becomes fundamental to help and control the process of learning, since this view extends learning beyond the formal boundaries of the classroom and provides a virtual learning environment always available (i.e. anywhere and anytime) for consultation and knowledge sharing with a strong impact on understanding the social dynamics among peers.

The groups of learners within a LMS could form one or more social networks. Once these networks have been correctly identified, it is possible to study their structure with social network analysis (SNA) techniques which allow to uncover non trivial structures [12] and important features.

For the aforementioned reasons the evolution of learning [4] is going toward the definition of a social learning management system (Social LMS) which allows to provide a "complete learning environment" that takes into account the social elements (e.g. collaborating, networking and information sharing capabilities) to improve the practices of learning, for this reason, within these platforms, the social aspects become central for all the activities.

The advantages which can stem from the utilization of a Social LMS are due to the fact that it is possible to provide an easy and uniform academic experience with the help of peers. At present the social features implemented in the majority of the LMSs are very limited and for this reason none of them can be considered as a fully Social LMS; however most of them include messaging systems which can be considered as a embryonic social feature. This research is focused on analyzing the knowledge sharing modules of LMSs in order to gain insights about the communication within the platform among peers. For this reason, we decided to study the utilization of the messaging system of the LMS presently in use at the Università degli Studi di Milano-Bicocca, Italy. This LMS is an instantiation of Moodle, version 3.1. Although LA is mainly directed towards the students, it is also aimed at teachers in order to improve their general vision on how learners are studying, the success of their learning practices, etc. [5,10].

On the other hand the perspective of this research is (mainly) that of the administrators where the goal is to monitor the several LMS's functionalities for managing the governance of academic institutions in order to check how learners and academic staff interact with the LMS. The tools here provided could improve the decisional process of the policies (regarding both hardware and software) this in turn should provide to the users a LMS of good quality. In our vision, good quality refers to providing efficient and effective e-learning services with good performances (e.g. high speed) of usability.

The present paper is an extended version of a work presented at the DATA 2017 international conference [1], where the explanation of the results has been extended, new analysis regarding the received messages have been included and a new paragraph containing the description of the architecture of a dashboard for accessing the indicators has been added.

The methodology followed in this research is the following: we first defined quantities of interest based on the present literature on the subject. Following these needs, a mathematical implementation has been proposed. The formulae have been confronted with data, and the resulting patterns have been modeled with parametric functions in order to summarize the most interesting features. At the end, we introduced new visualization and fruition tools in the form of dashboards. The two utilization indicators here defined are: *specific utilization* and *popularity*, respectively.

The idea behind them is to analyse how widespread the usage a specific functionality of a LMS is. In detail, *specific utilization* is an indicator aimed at verifying how many users accessed to the LMS activities in respect to all the possible users, whereas *popularity* is an indicator aimed at analysing the real usage of the functionality referred to the e-learning community which accessed it.

For a better understanding of this last functionality we defined the *real utilization plot*, which helps in visualizing the distribution of the utilization among the users. The *real utilization plot* allowed to notice a similarity between the observed trends and power laws. For this reason, we fitted the data with analytic functions and compared the parameters thus obtained with the Zipf law (which presents resemblances with our experimental curves).

The paper is organized as follows. A review of the present status of the indicators for LMSs is presented in Sect. 2. Section 3 defines the indicators aimed at analysing the utilization of generic LMS activities, with the objective to tune the policies of governance for better managing the e-learning platform. Section 4 presents a case study where the indicators have been applied to analyse the message activity for the Moodle platform used at the Università degli Studi di Milano-Bicocca, Italy. Section 5 presents the dashboard that was developed to allow an optimal access to the indicators. Finally, in Sect. 6 the conclusions are stated.

2 Related Work

Many research projects have been addressing the social activity of the students and teachers on a LMS, this section is going to be devoted to provide a brief report of those works and to highlight the differences and similarities with the present paper.

XRay [11] is an important suite for learning analytics based on Moodle. This suite includes many statistical tools to control and make predictions about the behavior of the students. Our research is focused on the point of view of the administrators rather than the students as in XRay. The particular indicators that we introduce, are not provided with the actual version of XRay.

The main goal of the paper presented in [9] is related to the problem of making predictions about the possible success of students in five classes of an online course (totaling 26 students) by collecting information of the underlying LMS. In order to better understand the dynamics within the students it is built a sociogram. A logistic regression is performed on top of the sociogram to provide a prediction of the success of the learners. In this respect the study is also aimed at helping the teachers, but at variance with our work it is focused on the single student rather than providing a global utilization view of the features of a LMS.

A survey of the data mining techniques which could be useful for analysing a LMS is provided in [13]. The implementation of some of the techniques to the Moodle suite is also provided.

An interesting work on the importance of social network analysis is presented in [12], the intent is to understand the structures within groups of students. A tool aimed at establishing the educational social networks based on the asynchronous interaction provided by forums is presented and tested. Also in this case the idea is not to obtain a tool for controlling the utilization of the message system in a whole LMS but rather to obtain the structures arising among the students.

An extended method to create social graphs due to the social interaction in both synchronous (chat) and asynchronous (forums) contexts within a LMS is proposed in [2]. This study also provides an approach to take into account the time evolution of the bonds between the students.

In [19] it is introduced a nice experiment where the learning management system for two courses is replaced by Facebook groups. Since Facebook is an extremely popular social tool, it becomes natural to try to understand if it can effectively replace a LMS. The analysis provided in [19] suggests that the features of a carefully crafted LMS are still superior in respect to the utilization of Facebook groups for the same purposes and some students are concerned about their privacy when using Facebook instead of a social media devoted specifically to the learning environment.

3 Definition of the Indicators

The number of functionalities/activities in modern LMSs (such as, chat room, messaging system, forum, etc.) is increasing over time. For this reason, one can expect that some of them become more popular than others. In this respect, the range of users which can access to the functionality comprises two main groups: students and academic personnel. The actions performed by the students on a LMS include obtaining new skills, sharing learning material, communicating with peers, etc.; whereas academic personnel includes teachers, university managers, and LMS administrators. Who provide contents for the lessons and manage the functioning of the school activities.

This implies that the e-learning community is very heterogeneous, and thus requires a set of specialized tools for every possible role. It is common in fact that a LMS is provided with monitoring systems which allow to control the different activities happening on the platform. For example, a student might be interested in his/her own grades, a teacher might be interested in the activity of the single student or a whole class within a single subject. This research proposes an approach to help the governance of an academic institution, where the utilization indicators of a LMS are naturally divided according to different features/parameters (such as courses, academic years, etc.) that allow to correlate the utilization with the structure of the courses. In order to avoid misunderstanding we stress that the term utilization is used in this article as a synonym of the amount of accesses to a given LMS functionality. This quantity can be analysed in detail according to particular needs. In this respect, we propose to consider the amount of accesses divided by the total number of possible users. This quantity has been called *specific utilization* (or, in short, *su*) and it is obtained with the formula:

$$su(a,t,\mathbf{p}) = \frac{\#\ \text{ of accesses } (a,t,\mathbf{p})}{\#\ \text{ of users who can access } (a,t)} \tag{1}$$

The *specific utilization* provides a direct insight of the diffusion of a specific LMS functionality among the users within a particular department/area/course of utilization (a), at a given time (t), according to one or more parameters indicated here with the vector ($\mathbf{p}$). Let's consider, for example, the message system available to the students of the LMS in use at the Università degli Studi di Milano-Bicocca. In this case the parameter ($\mathbf{p}$) of Eq. 1 refers to the fact that we want to distinguish between the sent messages and the received messages. We are also interested in distinguishing subsets of the whole community who can access to the functionality (e.g. males vs females, or particular roles within the university). The *specific utilization* indicator is not limited to social functionalities but can be implemented on all the possible activities that a LMS user can access to. The information required to calculate this quantity is a timestamp related to the access and an identifier of the person who performed the access. Binning the utilization within fixed time spans allows to set the time granularity of the information (the academic year is a very natural choice, but one can decide to follow shorter or longer time frames).

Although the *specific utilization* indicator provides a quick insight about the success of a functionality, it is important to consider that some activities (although accessible from the whole student population) might be aimed specifically to a restricted group of users. In this case it might be more interesting to obtain information about the utilization of those who really accessed to the functionality (while neglecting the information regarding those who could access to the functionality but for some reason did not do it, and would skew the resulting statistical properties of the indicator). A functionality relevant only for a small subset of the whole student population, could be successful but it might obtain a small *specific utilization* score because of the normalizing constant proportional to the whole student population. This is the case of the message system for the present case study (Moodle at Università degli studi di Milano-Bicocca) where all the enrolled students have access to it, but some departments have a very limited implementation of the platform and, as a result, it becomes essentially useless for the student to access message system.

The real usage of an activity in a LMS is thus an interesting quantity, which can be better understood with the help of what we called the *real utilization plot.* The idea of this plot is to display the distribution of the population utilizing a given functionality. For example, on the abscissae there is the number of accesses to the functionality, while on the ordinate is displayed the amount of students which used the activity that particular number of times. The distribution of the population returns valuable information about the success of the activity.

Let us consider a case of a functionality which does not require many accesses within the span of a year and thus which has a low *specific utilization* score; for example it could be the functionality related to the procedure of defining a student's exam plan. This functionality is expected to be subject to a limited number of accesses per student (but most of the students should access it). If the data utilization of this functionality shows that there is a large amount of students who accesses to this functionality tens of times, this might imply that the associated service does not provide clear indications on how to complete the procedure correctly and thus the students need many accesses before solving their problem. The same *specific utilization* could be obtained in a completely different scenario, like a social functionality of a LMS which allows the students to share information with their peers. In this case, if the vast majority of the students accesses this functionality a limited number of times it is reasonable to think that a critical number of users has not yet been reached and for this reason the functionality is not really working as a social binding mechanism. From the *real utilization plot* as detailed above, it becomes natural to extract the weighed average of utilization, which we call *popularity*:

$$popularity(a, t, \mathbf{p}) = \frac{\sum_{n=1}^{\infty} n \cdot U_n(a, t, \mathbf{p})}{\sum_{n=1}^{\infty} U_n(a, t, \mathbf{p})}, \tag{2}$$

where $U_n(a, t, \mathbf{p})$ is the number of users who accessed n times to the functionality according to the department (a), time (t) and (possibly) a set of features denoted as $(\mathbf{p})$. Notice that the sum runs over the number of accesses n, which ranges from 1 to infinity. This is not a problem since U_n is different from 0 only on a finite number of values.

We use as a constant to normalize the results the sum of the users who accessed the functionality. The meaning of the *popularity* indicators is to understand how much the functionality has been accessed by the real users and neglect those who had the possibility but did not utilize it.

The administrators of a LMS can exploit the *popularity* indicator to better allocate the resources of the system, in fact the average amount of resources for utilization times the *popularity* times the number of active users provides a good insight of the total amount of resources to be allocated, while the variation of the *popularity* in respect of a given time frame is useful to predict the change in resources which might be needed.

4 Implementation of the Indicators

A Social LMS can be considered the natural evolution of a LMS, however today the available data about the social interaction in an e-learning environment is scarce. For this reason, we resorted at applying the indicators defined in the previous sections to the data retrieved from the LMS actually in use at Università degli Studi di Milano-Bicocca, which is Moodle, version 3.1.3.

Moodle (based on social constructivism) is one of the most popular Learning Management Systems in use among universities (there are over 7000 sites in 233 nations which are based on it). Moodle is an open project under the GNU GPL license (which probably allowed it to become one of the most popular LMSs in the world). It is thought as a support tool for the creation and management of online courses. Most of the information collected by Moodle is in the form of a relational database; in this respect, by using MySQL we retrieved the relevant information to extract the indicators so far defined and we used R to analyze the data. One of the modules (functionalities) present in Moodle is the message one, and it allows the students to communicate among themselves, with the administrators and with the teachers. The message module can be thus conceived as a preliminary step in the direction of a Social LMS and for this reason it is the natural point where to start our analysis. This module is at disposal to all of the students of the Università degli Studi di Milano-Bicocca (which comprises about 35000 students per academic year, during the time of our analysis).

In this paragraph we will describe the application of the *specific utilization* indicator detailed in Sect. 3. The available data spans the three academic years 2013/2014, 2014/2015 and 2015/2016. The idea of behind this investigation is to help the governance and control of the university from the point of view of the administrators. The process of retrieving the department of each student was rather cumbersome: from each message, we were able to obtain the internal email of the sender, at this point it was possible to find all the courses where the student was enrolled. In the database, each course is associated with a "department/area", and thus it was possible to link at least one department to each of the courses. Unfortunately, some of the courses were shared between different departments and this could lead to uncertainty, i.e. whether a student belonged to one or the other department. At the time when this analysis was carried out, this uncertainty could not be avoided, and as a result, some students have been classified in more than one department. Since this problem affects a minority of the population (less than 5%) we included in our analysis all the possible students for each department, allowing for duplicates. This implies that all the results presented in this paper are subject to an error of the order of 5% in the quantification of the indicators.

Although it might not be obvious, it is important to notice that the acts of sending and receiving messages provide different information regarding the use of the message functionality. One important reason for this distinction is that, in Moodle, the students can send one-to-one messages only, however teachers and administrators can send also one-to-many messages (this option has obviously been introduced in order to create an easy notification system). In this respect we can state that there are two main purposes when utilizing the message system:

1. notification (one-to-many messages)
2. simple interaction (one-to-one messages)

Unfortunately there is no easy way to extract whether a message belongs to the one-to-many or to the one-to-one class; of course a direct analysis of the text, could allow us to make such a distinction but it was beyond the scope of this work to apply data mining techniques to the body of the messages. In the case of the *specific utilization* associated with the sent messages we take into account only those messages which were sent by the student population and remove those which are due to the academic staff/teachers. In this regard we want to use the *specific utilization* to understand the success of the message system among the students as a socialization tool.

At the beginning we will consider the results obtained from the *specific utilization* regarding the time distribution of the sent messages during the different months, in different years. In order to not display a rather complicated 3D picture with both of the messages and departments as free variables, we grouped together all the departments and we considered the time evolution only (Fig. 1). The academic year 2013/2014 was when Moodle has been introduced as the university LMS, and for this reason it was a still rather new tool for all the users. With the passing of the time, the users become more acquainted with the new LMS and in this respect it is no surprise that there has been an increase of *specific utilization* along the years.

When comparing the utilization results related with a single month, it can be noticed that for almost all the months there has been an increase from 2013 to 2016. The variation of utilization from year to year in many cases seems rather constant, however in some months there has been a sort of saturation and the increase has almost stopped, or worse in some cases during the last academic year under consideration there has been a decrease of utilization.

The received messages on the other hand can provide information from both a the social point of view and as notifications (in which they are just another tool to receive technical information). The first obvious difference which can be noticed (Fig. 2) about the sent and received messages by month is that the latter

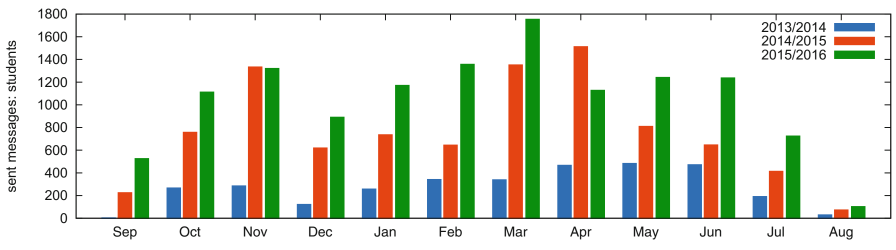


Fig. 1. Total amount of messages sent by month in the three academic years [1].

is about 5–6 times higher. Some features on the other hand are similar, also in this case there is a diffused increase of the number of received messages from one year to the next one, however this feature has many more exceptions in respect to the sent messages. For example, during the month of October 2014 more messages were received than the corresponding month of 2015, this being true also in February and March. There is also a case in which the first year of utilization surpasses the second one (see June of Fig. 2). When comparing the sent and received messages there is another anomaly which is very interesting. In the months of April associated with the academic year 2014/2015 and 2015/2016, in the case of the sent messages there is a decrease of utilization, while the opposite happens for the received messages which experience one of the highest increases over the three years.

The *specific utilization* associated with the sent messages among the different departments is shown in Fig. 3. The bars for each department refer to the three academic years under consideration. The two departments where the value of the indicator is higher are Sciences and Psychology. Almost all the departments are showing a steady increase of the value of *specific utilization*. On the other hand the absolute value of the *specific utilization* is generally very limited and it never exceeds 1. We also notice that Medical Sciences is the department with the lowest score of *specific utilization*.

If we take into account the received messages (Fig. 4), it is interesting to notice that the magnitude of the *specific utilization* is higher than the one of the sent messages (Fig. 3). In the case of the Sciences department (one of the departments with higher usage) the received messages outnumber the sent messages

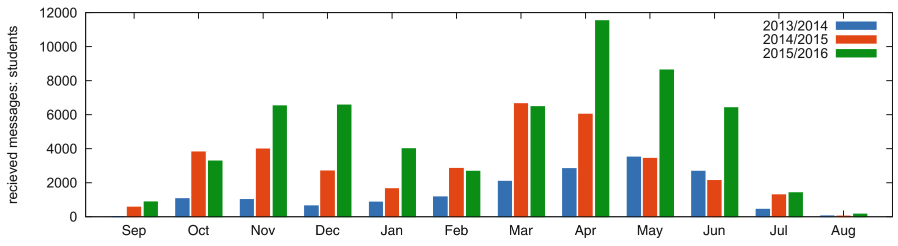


Fig. 2. Total amount of messages received by month in the three academic years.

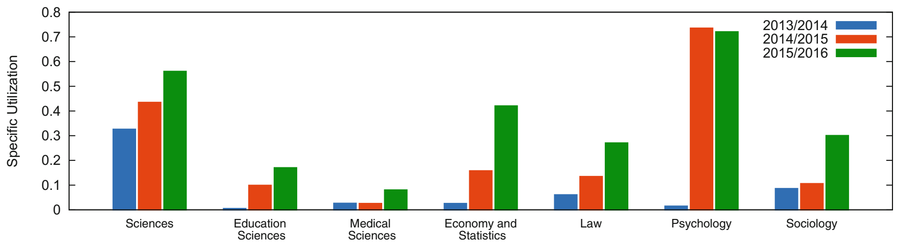


Fig. 3. Specific utilization of the sent messages among the students [1].

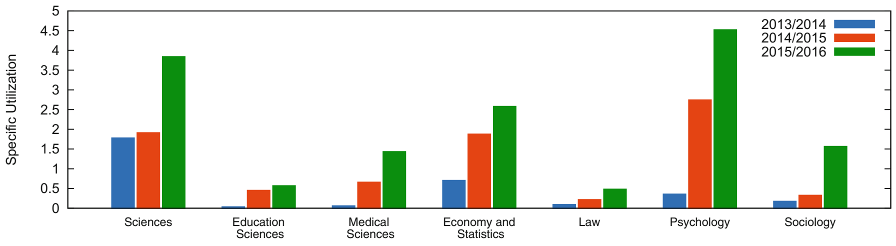


Fig. 4. Specific utilization of the messages received by the students.

more than 5 times for the academic year 2013/2014 and the gap increases over the years. Another interesting feature of this quantity in respect to the sent messages is that they show a rather poor correlation. For example in the case of Sciences, there has been a rather steady increase in the utilization of the sent message along the three academic years under consideration, while in the case of the received messages the highest increase has been registered passing from 2014/2015 to 2015/2016 when the *specific utilization* of the received messages has almost doubled passing from 2 to 4. This is not confirmed for the department of Economy and Statistics where the *specific utilization* of the sent messages has experienced the highest increase in the last year under consideration, at variance with respect to the received messages which in the corresponding period show the minimum increase. Even more curious is the case of the department of Psychology where there is an inversion. During the last year it has been registered a decrease of *specific utilization* of the sent messages and an increase of the one associated to the received messages.

There is an interesting feature to be noticed regarding the Psychology department since during the first year it had the second lowest score of *specific utilization*, while during the third year it jumped to the top position.

In this case we know that the internal regulations of the department forced all the teachers to migrate their courses over Moodle. In this respect, this is an example of the impact that policy regulations can have on the utilization of a feature. That being said, it is also fair to say that the policy regulation forcing a migration over Moodle had a limited effect if considered in terms of magnitude; in fact the *specific utilization* of the sent messages never exceeded the value of 1, which means that it is still used only by a niche of the total population which can access it.

The message functionality of a LMS is of key importance to establish a social network within the learning community. The opinion of the authors is that the social network of a LMS suffers the very strong concurrence of other means of communication [18] which are already well established among the students. In fact, even considering that all of the departments follow this increasing trend it would take many years to reach *specific utilization* values of the order of tens of messages per year. For this reason, the present status of the message system seems to require a qualitative change related to the LMS in order to reach a critical level.

In Figs. 1 and 2 there is a clear indication of a seasonal behavior, however, in order to have a solid statistical indication a few more years would have been needed, and an analysis should be carried out in future investigations. In particular during July, August and September the amount of exchanged messages is lower than the other months (due to the summer breaks), while March and April are typical exam session months which spark the need to exchange information and for this reason the utilization during these months is higher.

4.1 Popularity

This section considers the *popularity* associated with the sent messages. These messages have been divided in two parts, those sent only by the students and those by the academic personnel (which includes staff and teachers). It should be noted that, in the following analysis, it was not possible to make a distinction between teachers and administrative staff. This, in turn, implies that it is not possible to associate a given department with the senders which are labeled as academic. In order to compare the *popularity* of the students and of the academic personnel, we did not divide them between different departments either.

The *real utilization plots* which result in this case span many orders of magnitude in terms of users and of sent messages. As a result Figs. 5 and 6 have been displayed with a double logarithmic plot. As a contextualization it is important to know that the total number of student senders in the academic years 2013/2014, 2014/2015 and 2015/2016 is 9330, and the number of sent messages amounts to 24881.

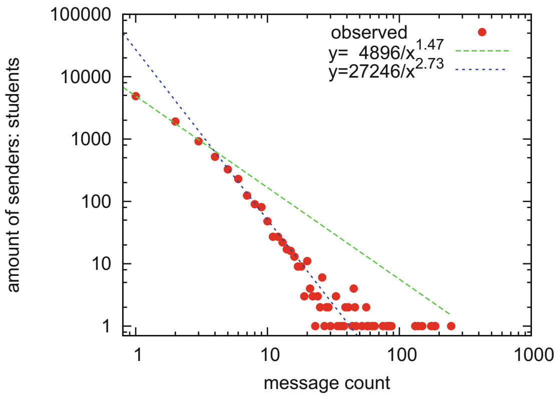


Fig. 5. Real utilization plot of the sent messages for the students [1].

As shown in the previous paragraph the *specific utilization* of this functionality of Moodle is scarce. Figure 5 shows the *real utilization plot* of the senders as a function of the number of messages sent during the three academic years.

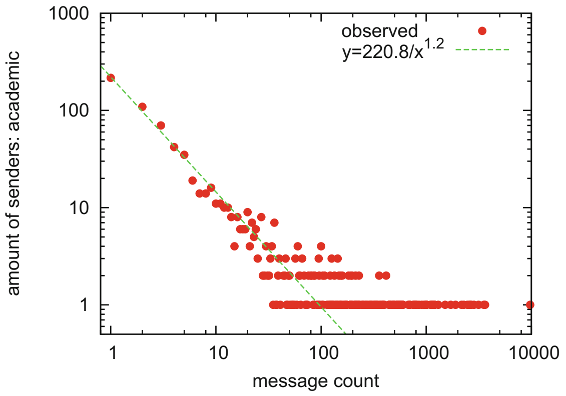


Fig. 6. Real utilization plot of the sent messages for the academic personnel [1]

In order to better explain these numbers we decided to fit the data with parametric families of functions. Although there is no obvious parametrization which can constrain all the features of the plot, power laws can catch some important points, in the following we will refer to functions of the form:

$$U_n = \frac{A}{n^k} \tag{3}$$

Where A is a constant, n represents the number of sent messages, while k involves the steepness of the power law (higher k implies a steeper descent as a function of n); U_n is the number of users who accessed n times to the functionality.

In Fig. 5 we can observe at least three different patterns. In the first part (between 1 and about 5 sent messages), the points are aligned, there is a kink in the distribution, while the points between 5 and 20 sent messages follow a straight line with a different slope in respect to the first ones. Above 20 messages is becomes difficult to consider the data as being produced by a simple parametrization. It should be taken into account that in this case there are many "exceptions", i.e. only one student sent 132 messages, while nobody sent 131 or 133 messages, which provides a staggering distribution for the tail.

As a result of a nonlinear least squares fit, the power law which best fits the first part of the graph has a coefficient $k = 1.5$ while A is 4896. It is interesting to remark that the value of A is essentially the number of students which sent just one message during the whole period of time comprising the three academic years. The parametrization which best fits the data between $n = 5$ and $n = 20$ is steeper, as a result the value of k is 2.73, and on the other hand $A = 27246$ (this would have been the amount of students sending just one message if all the points had followed this parametrization). As noted above the tail is due to many single persons who sent large amounts of messages. In particular in Fig. 7 we show the boxplot related to the messages sent by the students. It is clear that the median is exactly at one sent message, and those messages which are beyond

6 can be considered as being outliers, which means that these points are related with students who sent more than $\frac{3}{2}(Q_3 - Q_1)$ of the third quartile. In this case it is rather striking the fact that those students who sent 6 or more messages in three academic years can be considered as outliers in terms of sending many more messages than usual! In practice this includes about 8% of the active students.

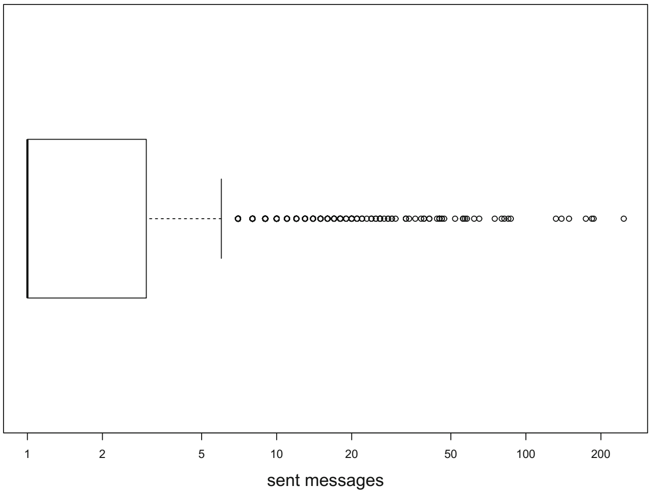


Fig. 7. Boxplot of the sent messages by the students.

The value of the *popularity* related to the student population is just 2.8. This confirms that also those who access to the message module do it very sporadically.

We expect that the utilization of the message module by the academic personnel should be rather different, this is due to the different roles of the teachers and the administrators but also because they can access to the one-to-many messages functionality. This possibility allows them to send notices to many people at the same time and thus the social aspect of the messages might not be the more important one, leaving room for a notification function. The amount of users is also very different; there are, in fact, 531 active senders and the total number of sent messages is 73357 (over the whole period of three academic years). In this case the *popularity* of the message system is 137, about 50 times higher than the *popularity* associated with the students. In Fig. 6, it is possible to notice that the data distribution does not show the same features of the student distribution, and a single power law can explain decently the points from $n = 1$ to $n \approx 40$. In this case the exponent k is equal to 1.2. As the number of sent messages increases the distribution becomes more and more noisy. Also in this case, above a certain number of sent messages there is a very noticeable tail. In this case however, it is responsible for shifting the value of the *popularity* to higher values. In Fig. 8 we show the boxplot corresponding to the academic personnel. In this case it can be noticed that the value of the median is 6 and that, in order for a member of the academic personnel to be considered as an outlier, he/she has to send more than 96 messages over the course of the 3 years. Those outlier cover about the 17% of the academic personnel.

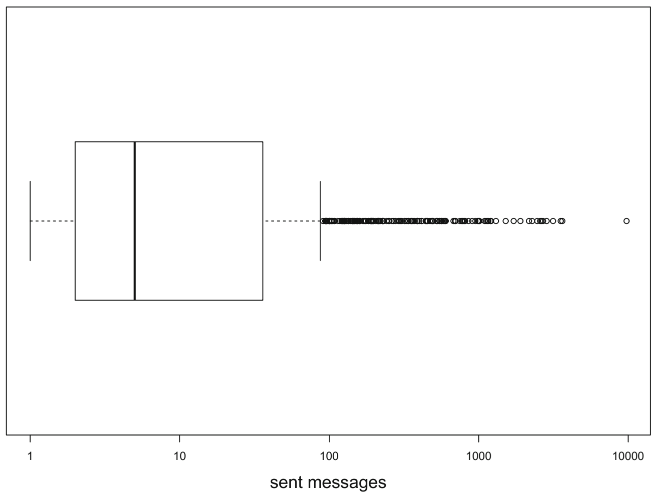


Fig. 8. Boxplot of the academic personnel sending messages.

In order to better compare the behaviors of the students and the academic personnel we decided to display them on the same graph. On the other hand this would be meaningless when considering the bare amount of users. For this reason we re-normalized the amounts of users by dividing by the total number of users who accessed to the message module (times 100 in order to obtain percentage values). With this re-normalization on the y-axes there is the percentage of students sending a given number of times messages. The two plots combined are shown in Fig. 9.

The usage due to the students shows shorter tails than the academic counterpart and also the number of users drops more quickly as a function of the number of sent messages. It is interesting to notice that around 20% of the academic personnel who accessed the message system did it only once, while this quantity raises to about 45% in the case of the students (this seems a clear indication of the fact that the academic personnel is more involved in the message system of the LMS). A confrontation of the shapes of tails is misleading. In the tails, there are single users who sent many messages but when re-normalized on the total population this returns different percentage values. Nonetheless it is striking that a large percentage of the population of the personnel belongs to the tails while the numbers are much smaller for the student population.

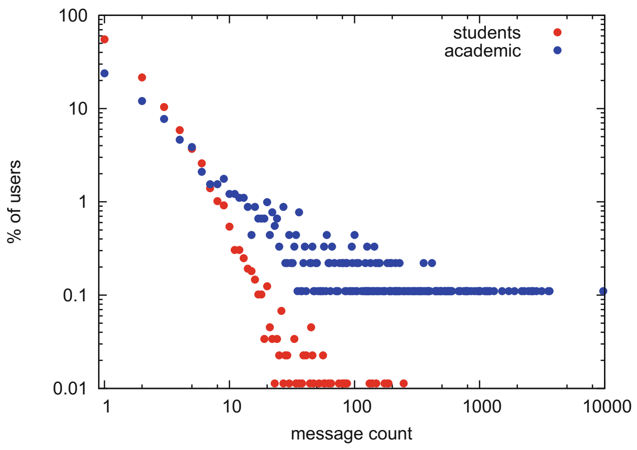


Fig. 9. The percentage of message senders as a function of the number of sent messages [1].

4.2 Zipf Law

The Moodle message module is thought to enhance communication among the users. In the very same area (communication among human beings), but in a rather different context, i.e. when quantifying the usage of the words in a text, there is a very well known phenomenon called Zipf law [20]. This relation has been discovered by studying the appearance frequency of the words in different texts. A very striking feature, which is present in a large percentage of texts, showing very little dependency on the language or the purpose of the text, is that the first most common term appears two times more frequently than the second most common term, and it is three times more common than the third most used term, and so on...

In detail the frequency of the n^{th} more common word is $1/n$ in respect to the most frequent term. In order to achieve a more general result it is possible to modify this formula where the appearance frequency f of the words (listed in order from the most common to the least one) read like:

$$\frac{f(1)}{f(n)} = n^k, \tag{4}$$

in detail, those parametrizations where the value of the parameter k is closer to 1 are more in line with the original formulation of the Zipf law. In this respect, the information gathered from the messages sent by the students seems to be rather distant from a Zipf law, for example because there are different features, a kink, long tails etc. (see Fig. 5)

The utilization of the message system by the academic personnel, however can be parametrized quite well with a single power law, where the coefficient $k = 1.2$. (see Fig. 6). In this respect it presents a curious resemblance to the original Zipf law.

The Zipf law has been associated with the principle of least effort [7], according to which humans tend to use the least effort if the result is acceptable for a given purpose. In this respect since there are easier means of communication it is reasonable that the students resorted at using the message module only when other systems were not feasible. The academic personnel, however, which does not have the same level of personal connection with the students was simplified by the features accessible via Moodle. This could be confirmed by the long tails, where, for the teachers/personnel, becomes easier to send one-to-many messages through the LMS rather than via normal email where they should input the name of each receiver.

In a successful message system (e.g. Facebook chat, Whatsapp, etc.) the information is naturally spread and enriched when passing from one person to the other. It is thus conceivable that the real utilization plot of a successful message system does not follow a Zipf-kind law, or at least that the exponent, associated with the descent in number of messages sent per person, should be very small (<1).

5 Dashboard

The present work is part of a broader project aimed at obtaining better tools for learning environments. As a result in this paper we are going to introduce the tools being developed to access to the indicators so far explained. The most prominent way to access to the indicators is in the form of dashboards. Dashboards, in fact, are very popular Learning Analytic tools for presenting data. Among the main features of dashboards we can find the customizability, i.e. the possibility to insert new indicators and functionalities by importing widgets. Dashboards can be used to record and display all the learning activities in order to promote self-awareness, considerations, and help the students to define goals and track their evolution. Dashboards for learning analytics can be associated with three different groups [23]:

1. The dashboards which support frontal lessons. These dashboards support the teachers to obtain feedback from the lessons they deliver, and allow them to adjust the classes according to the level of the students. For example the dashboard defined by [24] uses a hardware system to keep track of the interest of the students by analyzing their voice and their head movements.
2. Dashboards made to enhance and facilitate group works belong to the second group. For example TinkerBoard [25] tries to characterize the development of the activity of every group by quantifying the commitment.
3. Online-learning support dashboards. In this case the information obtained are designed to be visualized in order to promote discussions within the classroom. Most of the information is gathered from the log files produced by the LMSs.

AAT (Academic Analytics Tool) and X-Ray Analytics are two kinds of dashboards which belong to the third group of the aforementioned classification, since they extract data from the log files and they process them in order to provide useful information related to the engagement and performance of the students; in particular:

– AAT [21] is a dashboard connected to Moodle Analytics, which is a Moodle plugin. AAT was designed to be an easy access platform which enables to perform complex queries allowing to analyze the behavior of the students in the online courses. The users have a wide array of choices not limited to statistical indicators. The information extracted is mainly aimed at the analysis of single courses but it is possible to create more complex requests spanning more courses at the same time.
– X-Ray Analytics [22] is an application which enables to make predictions based on the information collected through MoodleRooms. It allows to analyze the trend which have an impact on the progresses and final results of the students. It allows to understand the learning behaviors in order to improve school performances and reduce the risk of failure ahead of time.

The dashboard produced for this project is different in respect to the ones just described in that it is focused on analyzing the social interactions of the users. These new characteristics of the users are more focused on soft skills rather than hard skills and for example include indicators of influence (which does not necessarily coincide with school skills). Each indicator is developed following a micro service strategy which allows the dashboard to be modular and adaptable to all the possible Technologies (it is thus not confined just to Moodle).

The project of this dashboard is based on three logical levels as shown in Fig. 10.

1. The first level is related to the storage, where data coming from different sources is analysed. Since the data can be defined in different ways it is required to make a unified representation model obtained with an intermediate level called ETL (Extraction, Transformation, Loading). The output is then saved in a data warehouse.
2. The second level includes data analysis and the definition of the indicators by using R and Python. The HRMS component (Handler Requests Micro Services) manages the requests of the client via restful calls, extracting the data of the requested indicators and later rewritten in JSON format. This last format is then sent to the client which will interpret graphically the data.
3. The third level deals with the front end built on a web application. The framework that we chose is Angular2 which allows to implement web applications usable on all sort of devices, like smartphones, tablets, desktops and laptops. This framework supports the programming language called Typescript. The modern aspect is obtained with material design styles. The library Highcharts is responsible for the interactive graphs within the dashboard.

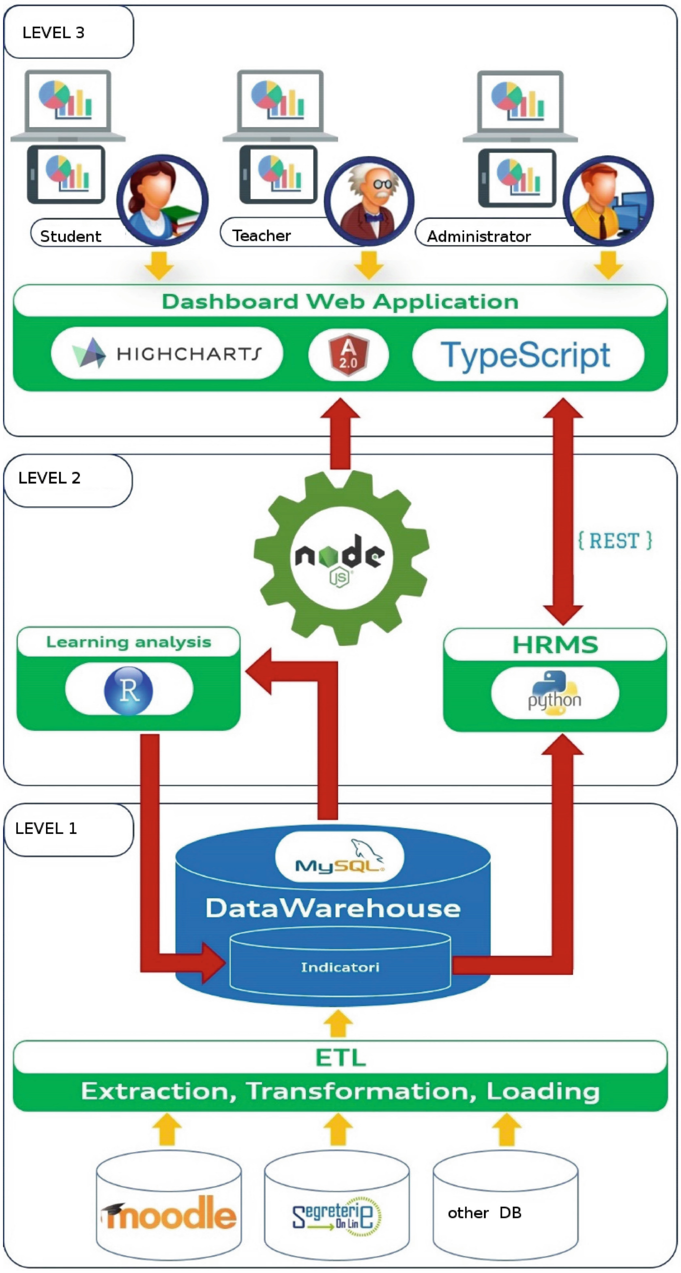


Fig. 10. Architecture of the dashboard.

6 Conclusions

The aim of this research is to provide new tools for analysing the amount of accesses to the functionalities of a LMS. In particular we are interested in assessing the utilization of those features which can potentially lead to a Social LMS where the information provided by the students plays a central role in the learning process. Since the social features of such a system might require a careful allocation of resources, this paper is mainly addressed at the administrators of the LMS who need to be in full control of the needs of the system. For this reason, two main indicators have been presented, *specific utilization* and *popularity*. The first one refers to the average utilization of a functionality among all the possible users, while the second one is more specific and it grants an insight about the detail of the real utilization by the users.

After defining the indicators, we tested them by analysing the data obtained by the LMS presently in use at the Università degli studi di Milano-Bicocca, which is an instance of Moodle (version 3.1). This LMS cannot be considered as a Social LMS since it is not centered around the social activity of the students, however there are social functionalities, like the message system. With the help of the newly defined indicators it was possible to confirm that the message functionality has not yet reached a critical stage in which there are active groups which create a self-sustained community. The development over time of the utilization has also been addressed and an increase of the values returned by the indicators has been reported. A similar check, related to the different departments has also been done, indicating that there are some departments which are more active than others, in particular, in one case this was attributed to a change of utilization policy defined at academic level. Even when taking into account the increase of *specific utilization*, in the near-medium term there is no foreseeable intense utilization of the message system, even worse, there are possible hints that the present stage of utilization might start to enter into a saturation phase soon (e.g. a slight decrease of utilization from one year to the next, related to particular months).

After considering the accesses as a whole, it was interesting to check in detail the real utilization of the system, with the help of the second indicator here defined. However, the value of the *popularity* of the message system (senders) among the students is also rather scarce (being around 3). A different perspective can be obtained by looking at the same functionality from the point of view of the academic personnel, where the *popularity* reaches a value of 137. By exploiting the *real utilization plot* it was possible to understand the detail of the noticeable difference between the students and the academic personnel. In the case of this second group of people the tails of the distribution of the users are responsible for this higher value of *popularity*. This is due to the fact that the academic personnel can send one-to-many messages and thus this functionality as a notification tool rather than a social tool.

Although the acts of sending and receiving messages might appear very similar they can carry rather different information. The results for *specific utilization* related to the received messages from the student population are in fact about 4–5 times higher than those related to the sent messages.

The distributions which appear in the *real utilization plot*, span many orders of magnitude and since they are essentially aligned (on double logarithmic plots), it appeared natural to parametrize them with power laws. This behavior is very similar to the empirical Zipf law which accounts for the appearance frequency of the words within a text. This law has been linked to the principle of least effort, where a person tends, provided a result is obtained, to employ the least possible energy in accomplishing it. In this respect one has to take into account that, nowadays, there are plenty of social tools which are extremely popular among the student population, and which are more easily accessible than the message system provided by Moodle. The users of the message system are reasonably more likely employing those tools when they need to communicate. The academic population, on the other hand, is facilitated by the Moodle message system when it is used as notification tool and for this reason the *popularity* of the messages is much higher in this group.

One of the problems related with the creation of a strong educational social network is the fact that the communication medium is limited in time by the enrollment of the students which follows the natural development of their career.

In an approach where a strong community is considered an important asset for obtaining a better education process it seems reasonable to suggest important changes of perspective when designing a Social LMS. In particular, an integration with existing social networks might benefit in terms of allowing the students to access to a familiar social feature.

The general approach of using indicators to control the development of a LMS requires means of visualization and access to the result. For this reason we introduced also the dashboard which is being developed. One of the natural properties of this dashboard is customizability, which in this case is achieved by using a modular approach where a user can define and implement the desired indicators by a simple action with the mouse.

References

1. Meluso, F., Avogadro, P., Calegari, S., Dominoni, M.: Success of the functionalities of a learning management system. In: DATA 2017 6th International Conference on Data Science, Technology and Applications, 24–26 July 2017, Madrid, Spain (2017)
2. Avogadro, P., Calegari, S., Dominoni, M.: Analyzing social learning management systems for educational environments. In: Fred, A., Dietz, J.L.G., Aveiro, D., Liu, K., Filipe, J. (eds.) IC3K 2015. CCIS, vol. 631, pp. 470–491. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-52758-1_25
3. Avogadro, P., Calegari, S., Dominoni, M.: Expert students in social learning management systems. Interact. Technol. Smart Educ. **13**(3), 202–217 (2016)
4. Dalsgaard, C.: Social software: e-learning beyond learning management systems. Eur. J. Open Distance E-Learning **2**(1), 1–9 (2006)
5. Ferguson, R.: Learning analytics don't just measure students progress they can shape it (2014). theguardian.com
6. Filiz, O., Kurt, A.: Flipped learning: misunderstanding and the truth. J. Educ. Res. **5**(1), 215–229 (2015)

7. Kingsley, G.: Human Behavior and the Principle of Least Effort. Addison-Wesley Press, Cambridge (1946)
8. Lukarov, V., Chatti, M.A., Schroeder, U.: Learning analytics evaluation - beyond usability. In: Proceedings of DeLFI Workshops 2015 Co-located with 13th e-Learning Conference of the German Computer Society (DeLFI 2015), Seiten/Artikel-Nr: 123–131 (2015)
9. Macfadyen, L.P., Dawson, S.: Mining LMS data to develop an early warning system for educators: a proof of concept. Comput. Educ. **54**, 588–599 (2010)
10. Mödritscher, F., Andergassen, M., Neumann, G.: Dependencies between e-learning usage patterns and learning results. In: Lindstaedt, S., Granitzer, M. (eds.) Proceedings of the 13th International Conference on Knowledge Management and Knowledge Technologies (i-Know 2013), article 24, 8 pages. ACM, New York (2013)
11. Moodlerooms. XRay learning analytics 2017. https://www.moodlerooms.com/resource/x-ray-learning-analytics/. Accessed 26 Apr 2017
12. Rabbany, R., Takaffoli, M., Zaïane, O.R.: Analyzing participation of students in online courses using social network analysis techniques. In: Proceedings of the 4th International Conference on Educational Data Mining, Eindhoven, The Netherlands, 6–8 July 2011 (2011)
13. Romero, C., Ventura, S., García, E.: Data mining in course management systems: Moodle case study and tutorial. Comput. Educ. **51**(1), 368–384 (2007)
14. Scheffel, M.: A framework of quality indicators for learning analytics. Learn. Anal. Rev. no. 2 (2015). ISSN 2057-7494
15. Scheffel, M., Drachsler, H., Specht, M.: Developing an evaluation framework of quality indicators for learning analytics. In: Proceedings of the Fifth International Conference on Learning Analytics and Knowledge LAK 2015, pp. 16–20. ACM Press, New York (2015)
16. Scheffel, M., Drachsler, H., Stoyanov, S.: Quality indicators for learning analytics. Educ. Technol. Soc. **17**(4), 117–132 (2014)
17. Sclater, N., Peasgood, A., Mullan, J.: Learning analytics in higher education. Jisc (2016)
18. Susilo, A.: Exploring facebook and whatsapp as supporting social network applications for English learning in higher education. In: PDE Professional Development in Education Conference 2014, 11–12 June 2014, Park Hotel Bandung (2014)
19. Wang, Q., Woo, H.L., Quek, C.L., Yang, Y., Liu, M.: Using the facebook group as a learning management system: an exploratory study. Comput. Educ. **54**, 588–599 (2010)
20. Zipf, G.K.: The Psychobiology of Language. Houghton-Mifflin, Oxford (1935)
21. Graf, S., Ives, C., Rahman, N., Ferri, A.: AAT: a tool for accessing and analysing students' behavior data in learning systems. In: Proceedings of the 1st International Conference on Learning Analytics and Knowledge (LAK 2011), pp. 174–179. ACM, New York (2011)
22. Website (2016). https://www.moodlerooms.com/resource/x-ray-learning-analytics/
23. Verbert, K., Govaerts, S., Duval, E., Santos, J., Van Assche, F., Parra, G., Klerkx, J.: Learning dashboards: an overview and future research opportunities. Pers. Ubiquit. Comput. **18**(6), 1499–1514 (2014)
24. Yu, Y.-C., You, S.-C.D., Tsai, D.-R.: Social interaction feedback system for the smart classroom. In: Proceedings of the 2012 IEEE International Conference on Consumer Electronics (ICCE), pp. 500–501. IEEE (2012)
25. Son, L.H.: Supporting reflection and classroom orchestration with tangible tabletops. Ph.D. thesis. http://biblion.epfl.ch/EPFL/theses/2012/5313/5313_abs.pdf

Experiences in the Development of a Data Management System for Genomics

Stefano Ceri(✉), Arif Canakoglu, Abdulrahman Kaitoua, Marco Masseroli, and Pietro Pinoli

Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano, Italy
{stefano.ceri,arif.canakoglu,abdulrahman.kaitoua, marco.masseroli,pietro.pinoli}@polimi.it

Abstract. GMQL is a high-level query language for genomics, which operates on datasets described through GDM, a unifying data model for processed data formats. They are ingredients for the integration of processed genomic datasets, i.e. of signals produced by the genome after sequencing and long data extraction pipelines. While most of the processing load of today's genomic platforms is due to data extraction pipelines, we anticipate soon a shift of attention towards processed datasets, as such data are being collected by large consortia and are becoming increasingly available.

In our view, biology and personalized medicine will increasingly rely on data extraction and analysis methods for inferring new knowledge from existing heterogeneous repositories of processed datasets, typically augmented with the results of experimental data targeting individuals or small populations. While today's big data are raw reads of the sequencing machines, tomorrow's big data will also include billions or trillions of genomic regions, each featuring specific values depending on the processing conditions.

Coherently, GMQL is a high-level, declarative language inspired by big data management, and its execution engines include classic cloud-based systems, from Pig to Flink to SciDB to Spark. In this paper, we discuss how the GMQL execution environment has been developed, by going through a major version change that marked a complete system redesign; we also discuss our experiences in comparatively evaluating the four platforms.

Keywords: Genomic computing · Data translation and optimization
Cloud computing · Next generation sequencing · Open data

1 Introduction

Thanks to Next Generation Sequencing, a recent technological revolution for reading the DNA, a huge number of genomic datasets have become available [23]. Massive pipelines are used to extract processed datasets from DNA sequences, and expose heterogeneous genomic and epigenomic signals; among them, TCGA (The Cancer Genome Atlas) [28], ENCODE (the Encyclopedia of DNA Elements) [14], and 1000 Genomes [1]. Open datasets of processed data are collected by worldwide consortia; they constitute a wealth of information, as they can be used for answering complex biological and clinical queries. While most of the processing load of today's genomic platforms is due to data extraction pipelines (e.g., [22]), we anticipate soon a shift of attention towards processed datasets, as such data are becoming increasingly available, collected by large consortia and by commercial sequencing centers. Genomics is expected to produce more data than any other scientific discipline by 2025, and is also expected to produce more data than all the social sources, including YouTube [26].

Processed datasets are used in *tertiary data analysis* for giving a global sense to heterogeneous genomic and epigenomic signals; a few systems are dedicated to tertiary data analysis, including SciDB Paradigm4 [3], and BLUEPRINT [2].

In the context of the GenData 2020[1] and GeCo Projects[2], we developed the Genomic Data Model (GDM) [20], a unifying model for processed data formats, and the GenoMetric Query Language (GMQL) [17,19], a query language for genomics. GDM is a very simple data model which mediates all existing data formats; GMQL is a high-level, declarative query language which supports data extraction as well as the most standard data-driven computations required by tertiary data analysis.

GMQL is the core of several, subsequent implementations of data management systems. In all cases, GMQL queries were translated to queries for cloud-based database engines. Version 1, described in [11], was developed between the spring 2014 and the spring 2015 and was based on Apache Pig [6,21] and Hadoop 1 [27]. Version 2 is described in [17]; its development started in the summer of 2015 and is still ongoing; Version 2 is based on Hadoop 2 [15] and uses Apache Spark [7,29]; project branches were developed for the engines Apache Flink [4] and SciDB [3,24]. In this paper, we discuss the development of the various GMQL versions, including the architectural choices, the supported optimizations, and the approaches to parallelism.

2 GMQL Resources: Data Model, Query Language, Integrated Repository

GMQL is based on a representation of the genomic information known as Genomic Data Model (GDM). Datasets are composed of samples, which in turn contain two kinds of data:

[1] http://www.bioinformatics.deib.polimi.it/gendata/, PRIN Italian National Project, 2013–2016.

[2] Data-Driven Genomic Computing, http://www.bioinformatics.deib.polimi.it/geco/, ERC Advanced Grant, 2016–2021.

– Genomic region values (or simply regions), aligned w.r.t. a given reference, with specific left-right ends within a chromosome: Regions of the model describe processed data, e.g., mutations, expression or bindings; they have a schema, with 5 common attributes (id, chr, left, right, strand) including the id of the region and the region coordinates, along the aligned reference genome, and then arbitrary typed attributes. This provides interoperability across a plethora of genomic data formats;
– Metadata, storing all the knowledge about the particular sample, are arbitrary attribute-value pairs, independent from any standardization attempt; they trace the data provenance, including biological and clinical aspects.

Figure 1 shows a GDM dataset consisting of 3 samples, each associated with a patient affected by Breast Cancer (BRCA). The region part has a simple schema, describing a particular value associated to each region; regions are aligned along the whole genome and belong to specific chromosomes (not shown in the figure). The metadata part includes free attribute-value pairs, describing the patient's age and sex; in GDM attributes are freely associated to samples, without specific constraints.

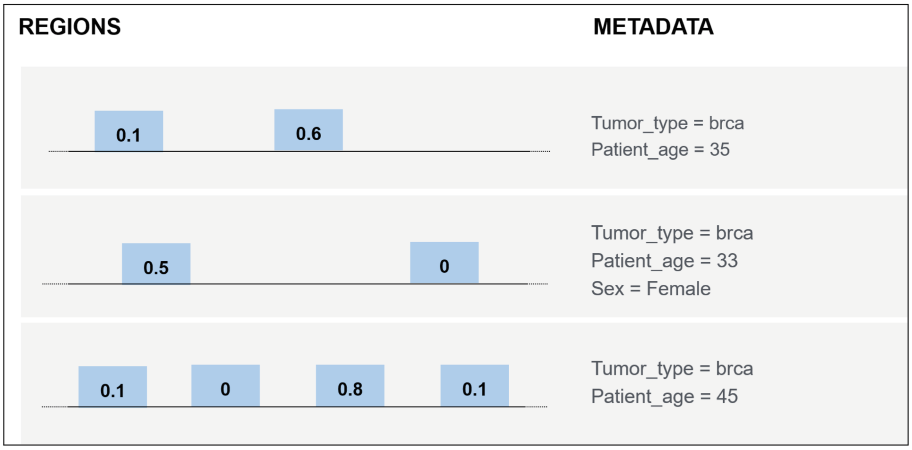


Fig. 1. Genomic Data Model.

GMQL is an abbreviation for GenoMetric Query Language - the name highlights the language's ability of computing distance-related queries along the genome, seen as a sequence of positions. GMQL is a closed algebra over datasets: results are expressed as new datasets derived from their operands. Thus, GMQL operations compute both regions and metadata, connected by sample identifiers; they perform schema merging when needed. The language brings to genomic computing the classic algebraic abstractions, rooted in Ted Codd's seminal work, and adds suitable domain-specific abstractions. In [19] we show GMQL at work in many heterogeneous biological contexts.

Table 1. Informal description of GMQL operations.

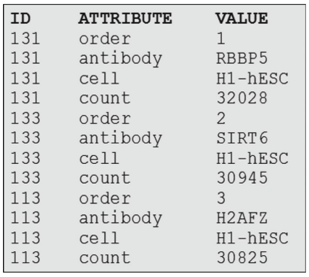
Operation		Description
Select	Meta	Selects complete samples when a metadata predicate is true (predicate can refer to other datasets using a semijoin condition)
	Regions	Selects regions satisfying a region predicate
Project	Meta	Projects metadata or adds/updates metadata attributes
	Regions	Projects regions or adds/updates region attributes
Extend	Meta	Adds to metadata aggregate function results computed over regions
Order	Meta	Sorts samples and selects top ones
	Regions	Sorts regions and selects top ones
Group	Meta	Groups samples and computes aggregate values
	Regions	Groups regions and computes aggregate values
Merge	Meta	Merges all metadata of all samples
	Regions	Merges all regions of all samples
Cover		Merges all samples; regions must satify an accumulation constraint
Union		Builds the union of samples (meta and regions independently)
Difference	Meta	Metadata of results are from the first operand
	Regions	Regions of second operand are subtracted from first operand
Map	Meta	Metadata of results are from the second operand
	Regions	Aggregates values of second operand's regions intersecting a region of the first one
Join	Meta	Builds pairs of samples whose metadata satisfy join predicate
	Region	Builds pairs of regions satisfying distal and equi-join predicates

GMQL operations include classic algebraic transformations (SELECT, PROJECT, UNION, DIFFERENCE, JOIN, SORT, AGGREGATE) and domain-specific transformations (e.g., COVER deals with replicas of a same experiment; MAP refers genomic signals of experiments to user selected reference regions; GENOMETRIC JOIN selects region pairs based upon distance properties); their semantics is informally described in Table 1. Tracing provenance both of initial samples and of their processing through operations is a unique aspect of our approach; knowing why resulting regions were produced is quite relevant.

An example of GMQL is the following, simple query which selects the genes' promoter regions from a dataset of annotations, selects al experiments of a given datatype from the NarrowPeaks dataset of ENCODE, and maps the selected peaks to the selected promoters; the query counts how many peaks intersect with each promoter region, then counts how many peaks in each sample intersect with any promoters, then selects the top thee samples based on the latter count. The counter is a global property of each sample and therefore is included by the EXTEND operation into the metadata of results. The language supports implicit iteration over all samples.

```
PROMS = SELECT(annotationType == 'promoter') ANNOTATIONS;
PEAKS = SELECT(dataType == 'ChipSeq') ENCODE_NarrowPeaks;
RESULT = MAP(peakCount AS COUNT) PROMS PEAKS;
GLOBAL = EXTEND(count AS SUM(peakCount)) RESULT;
TOP = ORDER(count; TOP 3) GLOBAL;
```

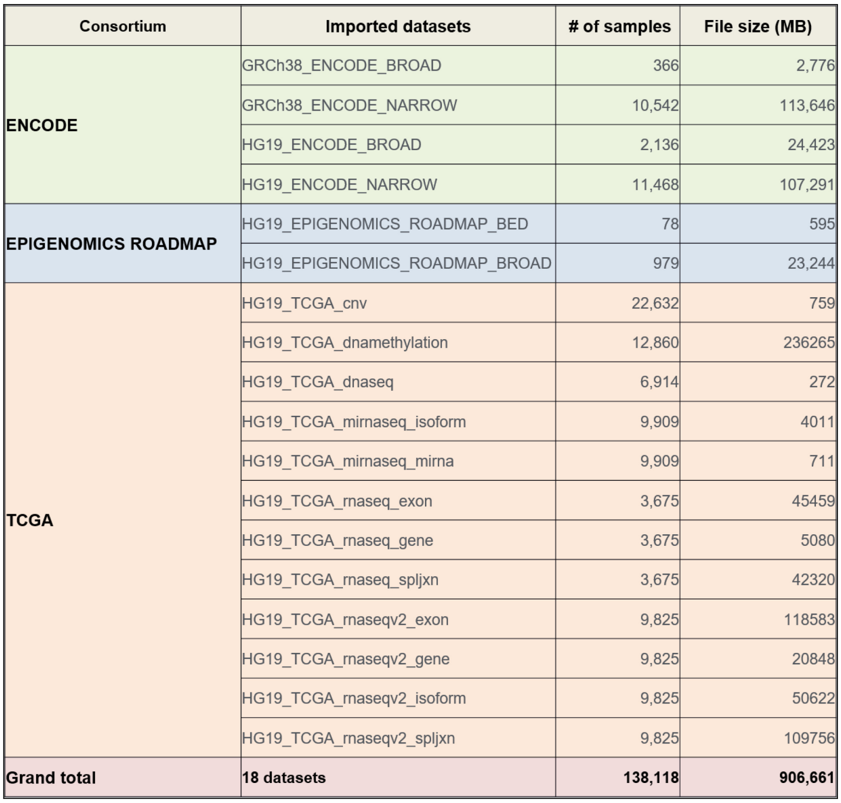
This query was executed over 2,423 ENCODE samples including a total of 83,899,526 peaks mapped to 131,780 human promoters, producing as result 29 GB of data. Figure 2 shows three samples with the highest counters (e.g., sample 131, corresponding to antibody target `RBBP5` of cell line `H1-hESC`, has 32,028 peaks intersecting with gene promoters.) Note that metadata of the result are partially computed by the query, partially derived from the initial description of experimental conditions in the `NarrowPeaks` dataset.



ID	ATTRIBUTE	VALUE
131	order	1
131	antibody	RBBP5
131	cell	H1-hESC
131	count	32028
133	order	2
133	antibody	SIRT6
133	cell	H1-hESC
133	count	30945
113	order	3
113	antibody	H2AFZ
113	cell	H1-hESC
113	count	30825

Fig. 2. Metadata of the query result.

In our current GMQL installation, we provide global curated datasets which have been converted to GDM, including all processed datasets available in TCGA [28] and ENCODE [14]; the transformation of TCGA datasets to GDM is discussed in [12]. The repository (as of January 1^{st} 2018) contains 18 datasets and 138,118 samples for a total size of 906 GB, as illustrated in Fig. 3; we are working towards its extension to many other source datasets, and the conceptual organization and integration of their most relevant metadata. We are also currently working on the transformation of datasets from The Genomic Data Commons (GDC) [30], which includes the latest version of TCGA datasets.



Consortium	Imported datasets	# of samples	File size (MB)
ENCODE	GRCh38_ENCODE_BROAD	366	2,776
	GRCh38_ENCODE_NARROW	10,542	113,646
	HG19_ENCODE_BROAD	2,136	24,423
	HG19_ENCODE_NARROW	11,468	107,291
EPIGENOMICS ROADMAP	HG19_EPIGENOMICS_ROADMAP_BED	78	595
	HG19_EPIGENOMICS_ROADMAP_BROAD	979	23,244
TCGA	HG19_TCGA_cnv	22,632	759
	HG19_TCGA_dnamethylation	12,860	236265
	HG19_TCGA_dnaseq	6,914	272
	HG19_TCGA_mirnaseq_isoform	9,909	4011
	HG19_TCGA_mirnaseq_mirna	9,909	711
	HG19_TCGA_rnaseq_exon	3,675	45459
	HG19_TCGA_rnaseq_gene	3,675	5080
	HG19_TCGA_rnaseq_spljxn	3,675	42320
	HG19_TCGA_rnaseqv2_exon	9,825	118583
	HG19_TCGA_rnaseqv2_gene	9,825	20848
	HG19_TCGA_rnaseqv2_isoform	9,825	50622
	HG19_TCGA_rnaseqv2_spljxn	9,825	109756
Grand total	**18 datasets**	**138,118**	**906,661**

Fig. 3. Repository of processed open datasets.

3 GMQL Implementation V1

The overall software architecture of GMQL V1 is shown in Fig. 4. From bottom to top, it includes the **repository layer**, the **engine layer** and the **GMQL layer**, which in turn consists of an **orchestrator** and a **compiler**, and is accessible through a **web service API**. We next briefly explain query execution, a detailed description can be found in [11]. Execution flow is controlled by the orchestrator, written in Java programming language; the processing flow includes compilation, data selection from the repository, scheduling of the Pig code execution over the *Apache Pig* engine [6], and storing of the resulting datasets in the repository in standard format.

When a user submits a GMQL query, the **orchestrator** calls the GMQL compiler, which produces the query translation into Pig Latin and the search criteria for loading the relevant samples from the repository; then, it uses the index manager to select from the repository the samples that comply to the search criteria. Then, it invokes the job optimizer, which sets the execution parameters (such as the parallelization factors); eventually, the orchestrator manages the outcome of the computation, including indexing of the result and storing it in the user space.

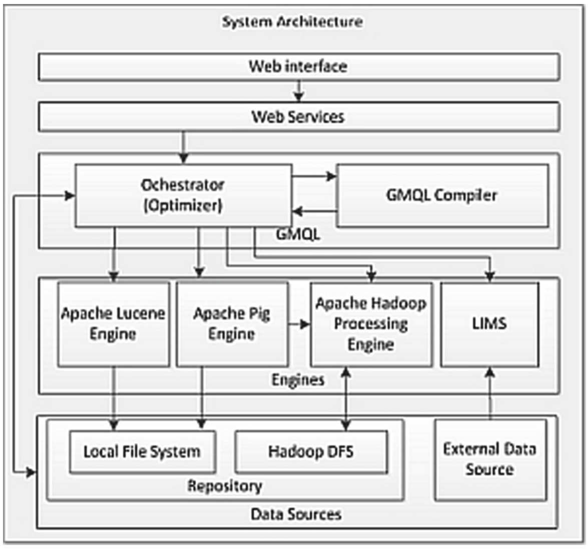
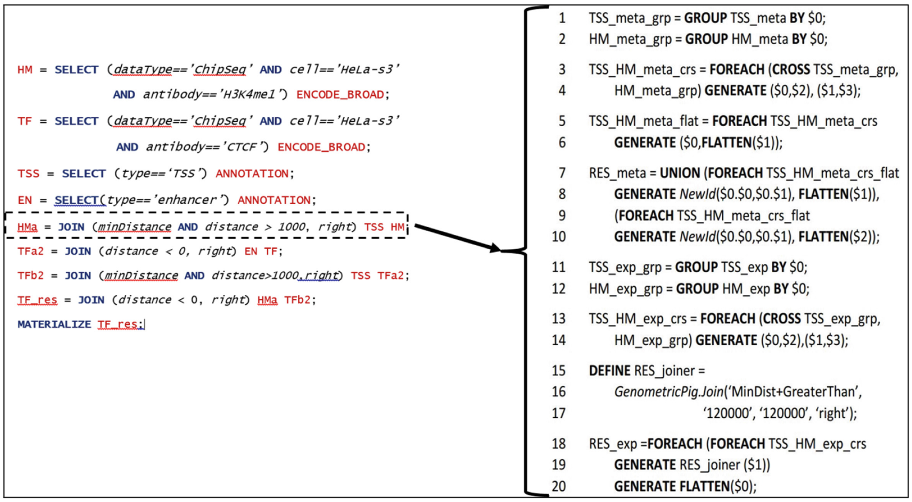


Fig. 4. Architecture of GMQL V1.

The **repository** includes a *Local File System* (LFS), organized within the Linux file system of the master node of the computing framework, and an *Hadoop Distributed File System* (HDFS) [25], shared among all the computing nodes. Datasets are stored in the HDFS system, subdivided in metadata and region data. Both the LFS and the HDFS store the control data, which include the schema for each dataset (encoded in Extensible Markup Language - XML) and the *Apache Lucene* [5] indexes for metadata. Moreover, both file systems have a public and a private space. Datasets are stored in their original text format; when they are selected by a GMQL query, they are serialized by suitable adapters and translated to the internal binary GDM format. At that point, they are managed by Apache Pig under Hadoop 1. In this way, we do not replicate data in the native and GDM formats and we minimize data translations from native into GDM format. Version 1 of GMQL was installed at IEO-IIT (http://genomics.iit.it/), a center of excellence in oncology research, with a direct connection to a Laboratory Information Management System (LIMS) designed for storing processed data as well as the raw datasets and the pipelines for their extractions.

The **compiler** analyses each GMQL statement by performing syntactic and semantic checks; for valid statements, it then infers the schema of the new introduced variable, then updates the internal state and finally emits the Pig Latin code that performs the requested operation. Datasets are loaded into suitable



```
HM = SELECT (dataType=='ChipSeq' AND cell=='HeLa-s3'
            AND antibody=='H3K4me1') ENCODE_BROAD;
TF = SELECT (dataType=='ChipSeq' AND cell=='HeLa-s3'
            AND antibody=='CTCF') ENCODE_BROAD;
TSS = SELECT (type=='TSS') ANNOTATION;
EN = SELECT(type=='enhancer') ANNOTATION;
HMa = JOIN (minDistance AND distance > 1000, right) TSS HM
TFa2 = JOIN (distance < 0, right) EN TF;
TFb2 = JOIN (minDistance AND distance>1000,right) TSS TFa2;
TF_res = JOIN (distance < 0, right) HMa TFb2;
MATERIALIZE TF_res;
```

```
1   TSS_meta_grp = GROUP TSS_meta BY $0;
2   HM_meta_grp = GROUP HM_meta BY $0;
3   TSS_HM_meta_crs = FOREACH (CROSS TSS_meta_grp,
4       HM_meta_grp) GENERATE ($0,$2), ($1,$3);
5   TSS_HM_meta_flat = FOREACH TSS_HM_meta_crs
6       GENERATE ($0,FLATTEN($1));
7   RES_meta = UNION (FOREACH TSS_HM_meta_crs_flat
8       GENERATE NewId($0.$0,$0.$1), FLATTEN($1)),
9       (FOREACH TSS_HM_meta_crs_flat
10      GENERATE NewId($0.$0,$0.$1), FLATTEN($2));
11  TSS_exp_grp = GROUP TSS_exp BY $0;
12  HM_exp_grp = GROUP HM_exp BY $0;
13  TSS_HM_exp_crs = FOREACH (CROSS TSS_exp_grp,
14      HM_exp_grp) GENERATE ($0,$2),($1,$3);
15  DEFINE RES_joiner =
16      GenometricPig.Join('MinDist+GreaterThan',
17                  '120000', '120000', 'right');
18  RES_exp =FOREACH (FOREACH TSS_HM_exp_crs
19      GENERATE RES_joiner ($1))
20      GENERATE FLATTEN($0);
```

Fig. 5. Translation of GMQL queries in V1.

Pig Latin variables; each dataset is mapped into two bags, named `V_region.dat` and `V_meta.dat`. The internal state contains the name and schema of each variable which is either generated or mentioned in the query. Figure 5 shows the translation of a `JOIN` operation, where lines 1–10 are concerned with metadata and produce the data bag `RES_meta`, and lines 11–20 work on regions by encoding in Java programming language a fast-join algorithm which searches for matching regions at minimal and bound distance. The linking of metadata and regions of each output sample is guaranteed by the use of the same hash function on the two `ids` of the input pairs, at lines 8 and 10 for the metadata and within the Join function for the regions.

3.1 Discussion

The development of GMQL V1 was relatively fast, as the implementation took about one year; Pig Latin was chosen as target language because GMQL syntactically recalls Pig Latin - both languages progressively construct queries by introducing intermediate results and naming them by using variables; this style of query writing was considered as similar to the scripting style which is dominant in bioinformatics, although of course GMQL scripts are high-level and not intertwined with programming language constructs.

The main problem with this approach is that the optimization strategies were hardcoded within the software produced by the translator, which in turn was focused on the specific features of our target language Pig Latin. An additional problem of V1 was the use of Lucene for metadata indexing; it turned out that access to metadata is much faster than access to regions, hence the reduction of execution time achieved by using Lucene at query load was not compensated by the need of preserving Lucene data structures aligned with the spontaneous evolution of the repository.

Moreover, at the end of 2014, it became clear that Apache Pig was no longer strongly supported, while other engines were becoming much more popular; in particular Spark was achieving a dominant position, Flink had very interesting capabilities as a streaming engine, and SciDB had a totally different approach to storage through arrays. Therefore, at the beginning of 2015, we opted for a major system redesign, starting the development of GMQL Version 2.

4 GMQL Implementation V2

The design of GMQL V2 has been centered around the notion of an intermediate representation for the query language, developed as an abstract operator tree (actually a directed acyclic graph or DAG, as operations can be reused), that is at the center of the software architecture. The intermediate representation carries the semantics of the query language in terms of elementary operations that are applicable to metadata and to region data separately, and opens up to various options for expressing the language's syntax (which can be expressed as relational expressions, or in embedded form within programming or workflow languages, or in logical format by using a Datalog-like style) and for deploying over different cloud-based engines. In particular, at various times we used Spark, Flink and SciDB.

The intermediate representation is also the backbone for supporting several language implementations which embed GMQL operations within a programming paradigm (Fig. 6). We have developed a library for Python (called `gmql`), currently registered within the standard Python libraries; it enables running GMQL queries both on a local repository (on the user's desktop) and on a global repository (e.g., the CINECA installation). We are also developing interfaces for R and Galaxy.

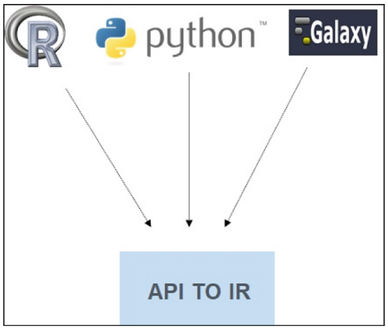
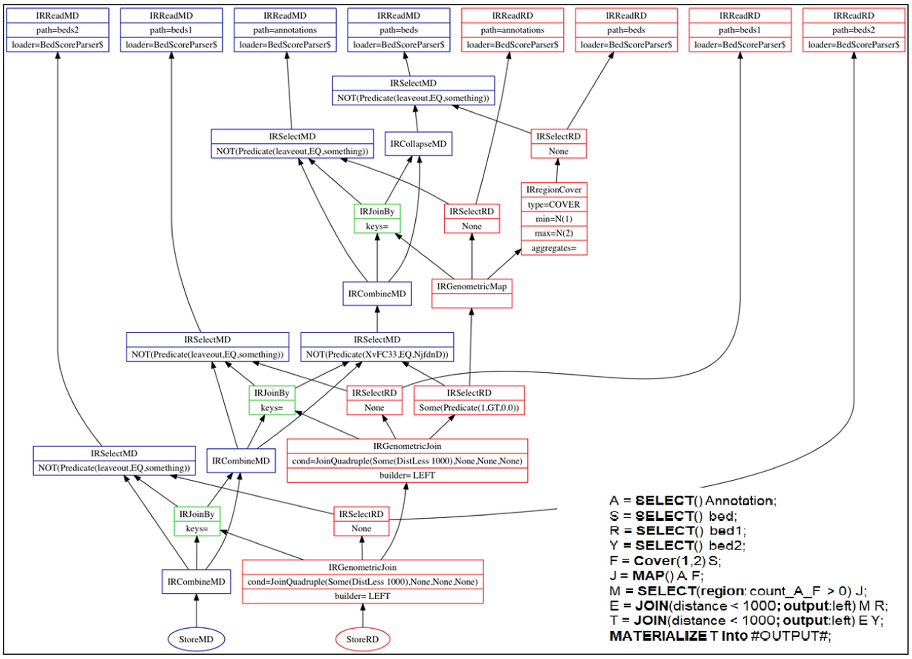


Fig. 6. Language interfaces for V2.

A DAG for a complex query is illustrated in Fig. 7. The query consists of five SELECT, two JOIN, one COVER and one MAP statements. It has four input datasets (called `Annotation, Bed, Bed1, Bed2`), each in turn represented by separate metadata and region data structures. Blue nodes apply to metadata, red nodes apply to regions. The orientation of edges indicates that the nodes are evaluated from the bottom to the top; indeed any evaluation occurs with the execution of the MATERIALIZE statement which indicates which variable of the program should be returned as result (in the specific example, variable `T`) and how the result variable should be named in the private repository of the user issuing the query (in this specific case, `#OUTPUT#`).



IRReadMD path=beds2 loader=BedScoreParser$
IRReadMD path=beds1 loader=BedScoreParser$
IRReadMD path=annotations loader=BedScoreParser$
IRReadMD path=beds loader=BedScoreParser$
IRReadRD path=annotations loader=BedScoreParser$
IRReadRD path=beds loader=BedScoreParser$
IRReadRD path=beds1 loader=BedScoreParser$
IRReadRD path=beds2 loader=BedScoreParser$
IRSelectMD NOT(Predicate(leaveout,EQ,something))
IRSelectMD NOT(Predicate(leaveout,EQ,something))
IRCollapseMD
IRSelectRD None
IRregionCover type=COVER min=N(1) max=N(2) aggregates=
IRJoinBy keys=
IRSelectRD None
IRCombineMD
IRGenometricMap
IRSelectMD NOT(Predicate(leaveout,EQ,something))
IRSelectMD NOT(Predicate(XvFC33,EQ,NjfdnD))
IRJoinBy keys=
IRSelectRD None
IRSelectRD Some(Predicate(1,GT,0.0))
IRSelectMD NOT(Predicate(leaveout,EQ,something))
IRCombineMD
IRGenometricJoin cond=JoinQuadruple(Some(DistLess 1000),None,None,None) builder= LEFT
IRJoinBy keys=
IRSelectRD None
IRCombineMD
IRGenometricJoin cond=JoinQuadruple(Some(DistLess 1000),None,None,None) builder= LEFT
StoreMD
StoreRD

```
A = SELECT() Annotation;
S = SELECT() bed;
R = SELECT() bed1;
Y = SELECT() bed2;
F = Cover(1,2) S;
J = MAP() A F;
M = SELECT(region: count_A_F > 0) J;
E = JOIN(distance < 1000; output:left) M R;
T = JOIN(distance < 1000; output:left) E Y;
MATERIALIZE T Into #OUTPUT#;
```

Fig. 7. DAG for a complex GMQL query.

The DAG shows that each GMQL operation is mapped to one or more operations on the meta and region datasets. It also shows that the subtree constituted by the blue nodes, which apply to metadata, can be computed before the subtree constituted by the red nodes, which apply to regions. This property hints to a powerful optimization, called *meta-first*, which only applies to a subset of the GMQL queries - by virtue of such optimization, it is possible to load only regions of the samples which are used to construct the result.

We next briefly describe the software architecture of V2; similarly to V1, the architecture includes as principal components a database engine and a global repository[3]. The software of V2 is organized according to a four layer architecture:

[3] GeCo V2 software is available at https://github.com/DEIB-GECO/GMQL.

- The **Access Layer** supports the various interfaces available in V2. These include an API to the DAG intermediate representation, a shell command line interface, and a collection of Web Services for accessing GMQL resources. Web Services are used for developing a user-friendly Web interface.
- The **Engine Components**, including the GMQL Compiler, for compiling a GMQL query into a DAG (which embodies execution plans); the DAG Manager, for supporting the creation and dispatching of DAG operations to other components; the Server Manager, for managing multi-user execution and their access capabilities; the Repository Manager, for managing the access to the repository; and the Launch Manager, for launching the executions of implementations.
- The **Implementation Components** (or executors), including the three implementations currently supported. Each package contains the implementation of the operations (abstract classes) of the DAG nodes, respectively for the Spark Implementation (the default option, and the most stable of the current implementations) and the Flink and SciDB implementations (which have been developed essentially for comparing their performances to the Spark implementation).
- The **Repository Implementations**, including a Local File System (LFS) repository, used when the installation is for a single machine, and a Remote File System (RFS), used in a cluster-based architecture. We currently support three installations, respectively at the CINECA Supercomputing center, on the Amazon Cloud, and on a dedicated local server available at our home institution.

The Repository Manager is the system component in charge of storing and managing the datasets imported from external repositories or generated by an user as result of a query execution. We support a private repository for each user and a public, read-only repository shared by all the users, which contains datasets from open public collections, such as ENCODE [14] and TCGA [28], as discussed in Sect. 2.

4.1 Optimization Options in V2

The new architecture opens up for multiple optimization options, illustrated in Fig. 8. In particular, a query optimizer applies deployment-independent optimizations directly to the intermediate representation, either in the form of node reordering or of refinements of the initial selections; the former ones range from classic algebraic optimizations, e.g., pushing of selection conditions to the nodes where metadata are used for joining or grouping (operations JOIN, MAP, GROUP, and ORDER), or distribution of selections to binary operations (such as JOIN, MAP, UNION and DIFFERENCE), or inversion of the execution order of binary operations (such as JOINS with UNION or DIFFERENCE); these optimizations are subject to applicability constraints. In few cases, it is possible to perform also node deletions, e.g., when after optimization we can infer that meta predicates are contradictions. These optimizations produce an Optimized Intermediate Representation which is next used for the GMQL implementations.

Other optimizations are possible at a low level, and apply to specific implementations. In Version 2, we associate each dataset with a **profile**, which includes several parameters, such as: the number of samples, their sizes, the number of regions in each sample and the length of the genome where sample regions are distributed (these are typically a subset of the overall length of each chromosome, as the initial and final parts of the chromosomes are not typically involved in protein coding or epigenomically relevant regions). Therefore, it is possible to use alternative algorithm for deploying the various operations depending on the profiles of operands, or to dedicate a suitable number of execution nodes to given queries based on their expected load, or to optimally use data partitioning and caching, so as to evenly distribute the load to the various nodes of a cloud-based system.

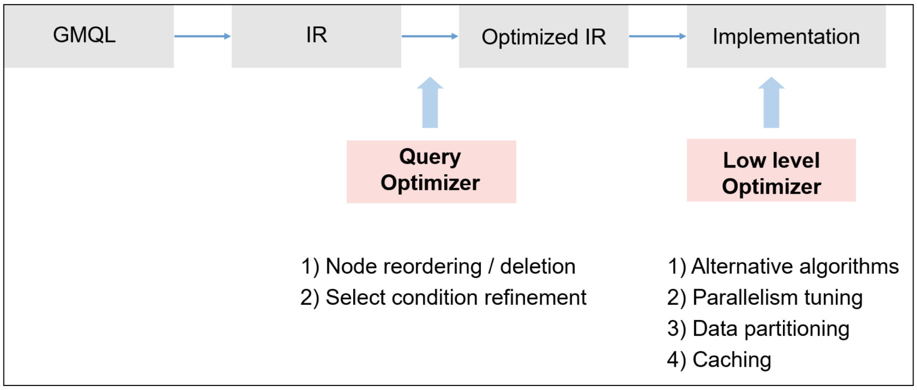


Fig. 8. Optimization options for GMQL V2.

The most computationally expensive operations in GMQL require joining regions of two datasets; each dataset may have millions of regions, yielding to operations which potentially range over thousand of billions of region pairs. For coping with such requirements we use genome binning, illustrated in Fig. 9; binning is effective in supporting both effective parallelization and reduction of the pairs of regions to be considered for each join predicate. Binning has been used in the context of genome browsers to speed up query processing over regions [18]; its use in the map-reduce computations was proposed in [13].

Genome binning consists of partitioning the genome into equal-size segments; each region of the two operands of a joining operation is assigned to one or more bins, and the joins between regions are executed only within bins, yielding very effective parallel processing. Specifically, the first operand of the join is called *anchor* and the second operand is called *experiment*; the mapping of each anchor region to bins is defined by its *search space*, which in turns depends on the genometric join condition. Experiment regions are simply mapped to bins on the basis of their position. Algorithms discussed in [17] describe the binning

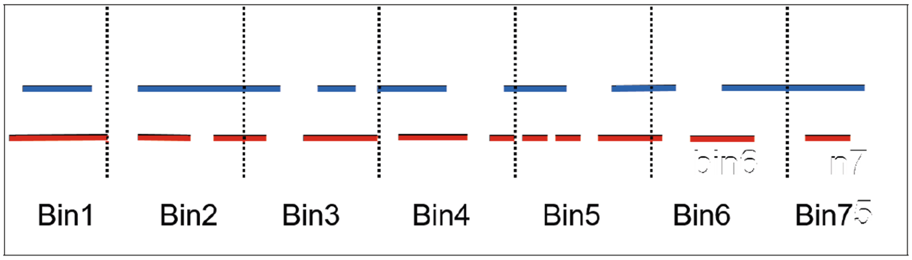


Fig. 9. Genome binning.

algorithms in detail and show that each operation is associated with an *optimal bin size*. Intuitively, small bin sizes produce an excess of parallel execution, while large bin sizes produce an excess of execution of join operations within each bin.

4.2 Performance Comparison

A direct comparison between V1 and V2 is not very significant, because the two systems use different technologies and have been deployed over different platforms.

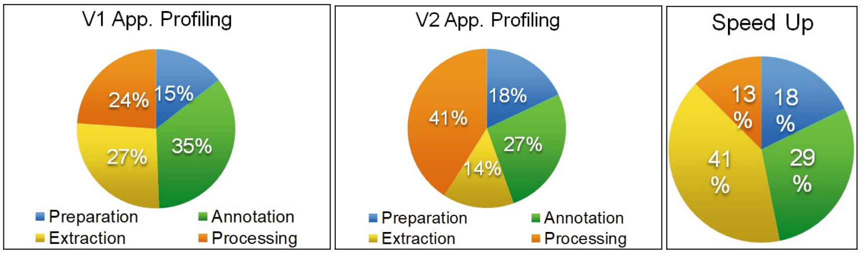


Fig. 10. Use of resources and speedup, V1 vs. V2.

A comparative study was done at the time of the first release of V2, on the same platform, using the Spark engine for V2; the performance of V2 has significantly increased since its first release. We considered four GMQL query fragments, specifically dedicated to data preparation (with SELECT and COVER as dominant operation), differential data annotation (with EXTEND and DIFFERENCE as dominant operation), processing (with SELECT and JOIN dominant operations) and extraction of results (with MAP dominant operation). The comparison between V1 and V2, shown in Fig. 10, shows the use of resources for each query fragment and the speed-up in going from Version 1 to Version 2, ranging from 13% to 41%.

Other comparative studies regard the speed-up of specific operations. The left diagram of Fig. 11 corresponds to a MAP operation with a single reference and shows that both V1 and V2 scale linearly with the number of samples, but V2 has about half execution time; the right diagram of Fig. 11 corresponds to a MAP operation with multiple references (N = 11) and shows that V1 does not scale linearly.

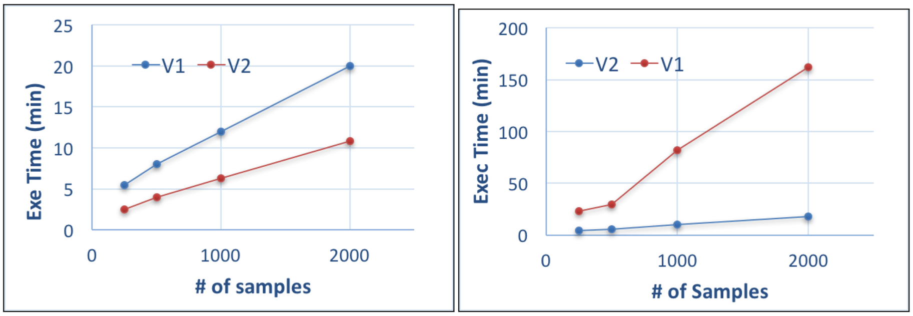


Fig. 11. Speedup of the MAP operation, V1 vs. V2.

4.3 Discussion

Compared to V1, V2 has a much more complex software organization. While V1 was cooperatively developed by 3 PhD students, V2 has a larger group of developers, which included in the past several master students; the use of GitHub allowed for several branches to the main Spark implementation, which allowed us to test also the use of Flink and SciDB, served by suitable implementation branches. At the moment, the Spark implementation is fully supported, while the Flink and SciDB implementations are only maintained for specific operations.

Comparative analysis, published in [9,10], shows that the performance of Flink and Spark are remarkably similar, while the performance of Spark and SciDB are very different, with SciDB faster then Spark when operations involve selections and aggregates (as they are facilitated by an array organization); whereas, Spark is faster than SciDB in JOIN and MAP operations (thanks to the general power of the Spark execution engine.)

5 Conclusions

The GMQL System is the center of several other tools and activities. We fully developed a Python library supporting an interface to the full GMQL language; the library is deployed within the international Python library repository and does not require any installation - as it is customary in Python. It supports a local and a remote execution mode, the former runs in the client desktop, the latter runs on a remote server and as such it connects to the global repository. Similar interfaces are being deployed for R and Galaxy.

We also developed a client-side system for data inspection and exploration, described in [16]; the tool connects directly to the results of GMQL queries and specifically to those queries which are completed by a MAP operation, as it is typically used to map known annotations (e.g., genes or epigenomically selected regions) to experimental datasets (e.g., gene expressions or mutations).

Our major current effort is dedicated to the development of an integrated repository for processed datasets, that extends the current repository described in Sect. 2. We are currently defining core metadata information that is normally available at all data sources, and then methods for extracting and normalizing such information; the conceptual model of the integrated repository is presented in [8]. All these efforts are coherently applied to tertiary data analysis, whose relevance is expected to grow within the near future.

Acknowledgment. This research is funded by the ERC Advanced Grant project GeCo (Data-Driven Genomic Computing), 2016–2021.

References

1. 1000 Genomes Consortium: An integrated map of genetic variation from 1,092 human genomes. Nature **491**, 56–65 (2012)
2. Albrecht, F., et al.: DeepBlue epigenomic data server: programmatic data retrieval and analysis of the epigenome. Nucleid Acids Res. **44**(W1), W581–586 (2016)
3. Anonymous paper, Accelerating bioinformatics research with new software for big data to knowledge (BD2K), Paradigm4 Inc. (2015). http://www.paradigm4.com/
4. Apache Flink. http://flink.apache.org/
5. Apache Lucene. http://lucene.apache.org/core/
6. Apache Pig. http://pig.apache.org/
7. Apache Spark. http://spark.apache.org/
8. Bernasconi, A., Ceri, S., Campi, A., Masseroli, M.: Conceptual modeling for genomics: building an integrated repository of open data. In: Mayr, H.C., Guizzardi, G., Ma, H., Pastor, O. (eds.) ER 2017. LNCS, vol. 10650, pp. 325–339. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69904-2_26
9. Bertoni, M., et al.: Evaluating cloud frameworks on genomic applications. In: Proceedings of IEEE Conference on Big Data Management, Santa Clara, CA (2015)
10. Cattani, S., et al.: Evaluating big data genomic applications on SciDB and Spark. In: Proceedings of Web Engineering Conference, Rome, IT (2017)
11. Ceri, S., et al.: Data management for heterogeneous genomic datasets. IEEE/ACM Trans. Comput. Biol. Bioinform. **14**(6), 1251–1264 (2016)
12. Cumbo, F., et al.: TCGA2BED: extracting, extending, integrating, and querying The Cancer Genome Atlas. BMC Bioinform. **18**(6), 1–9 (2017)
13. Chawda, B., et al.: Processing interval joins on Map-Reduce. In: Proceedings of EDBT, pp. 463–474 (2014)
14. ENCODE Project Consortium: An integrated encyclopedia of DNA elements in the human genome. Nature **489**(7414), 57–74 (2012)
15. Hadoop 2. http://hadoop.apache.org/docs/stable/
16. Jalili, V., et al.: Explorative visual analytics on interval-based genomic data and their metadata. BMC Bioinform. **18**, 536 (2017)

17. Kaitoua, A., et al.: Framework for supporting genomic operations. IEEE-TC (2016). https://doi.org/10.1109/TC.2016.2603980
18. Kent, W.J.: The human genome browser at UCSC. Genome Res. **12**(6), 996–1006 (2002)
19. Masseroli, M., et al.: GenoMetric Query Language: a novel approach to large-scale genomic data management. Bioinformatics **31**(12), 1881–1888 (2015)
20. Masseroli, M., et al.: Modeling and interoperability of heterogeneous genomic big data for integrative processing and querying. Methods **111**, 3–11 (2016)
21. Olston, C., et al.: Pig Latin: a not-so-foreign language for data processing. In: ACM-SIGMOD, pp. 1099–1110 (2008)
22. Roy, A., et al.: Massively parallel processing of whole genome sequence data: an in-depth performance study. In: ACM Sigmod, Boston, MA (2017)
23. Schuster, S.C.: Next-generation sequencing transforms today's biology. Nat. Methods **5**(1), 16–18 (2008)
24. SciDB. http://www.scidb.org/
25. Shvachko, K., et al.: The Hadoop distributed file system. In: Proceedings of MSST, pp. 1–10 (2010)
26. Stephens, Z.D., et al.: Big data: astronomical or genomical? PLoS Biol. **13**(7), e1002195 (2015)
27. Taylor, R.C., et al.: An overview of the Hadoop MapReduce HBase framework and its current applications in bioinformatics. BMC Bioinform. **11**(Suppl. 12), S1 (2010)
28. Weinstein, J.N., et al.: The Cancer Genome Atlas Pan-Cancer analysis project. Nat. Genet. **45**(10), 1113–1120 (2013)
29. Zaharia, M., et al.: Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In: Proceedings of USENIX, pp. 15–28 (2012)
30. Jensen, M.A., et al.: The NCI Genomic Data Commons as an engine for precision medicine. Blood **130**(4), 453–459 (2017)

Databases and Data Security

Server-Side Database Credentials: A Security Enhancing Approach for Database Access

Diogo Domingues Regateiro(✉), Óscar Mortágua Pereira, and Rui L. Aguiar

DETI, University of Aveiro, Instituto de Telecomunicações, 3810-193 Aveiro, Portugal
{diogoregateiro, omp, ruilaa}@ua.pt

Abstract. Database applications are a very pervasive tool that enable businesses to make the most out of the data they collect and generate. Furthermore, they can also be used to provide services on top of such data that can access, process, modify and explore it. It was argued in the work this paper extends that when client applications that access a database directly run on public or semi-public locations that are not highly secured (such as a reception desk), the database credentials used could be stolen by a malicious user. To prevent such an occurrence, solutions such as virtual private networks (VPNs) can be used to secure access to the database. However, VPNs can be bypassed by accessing the database from within the business network in an internal attack, among other problems. A methodology called Secure Proxied Database Connectivity (SPDC) is presented which aims to push the database credentials out of the client applications and divides the information required to access them between a proxy and an authentication server, while supporting existing tools and protocols that provide access to databases, such as JDBC. This approach will be shown and further detailed in this paper in terms of attack scenarios, implementation and discussion.

Keywords: Access control · Software architecture
Security and privacy protection · Network communications
Database connectivity

1 Introduction

Securing sensitive data is an important step when it can be access through a service provided by an organization. However, it is impossible to completely secure the data without making the system unusable in the first place, and making it overly secure would require many security checks. Therefore, the key lies in finding the right balance between usability and security. When it comes to databases, the usage of standard APIs to access and manipulate data, such as Java Database Connectivity (JDBC) [1], Hibernate [2] and other similar mechanisms is very pervasive, however they lack on the side of security.

When client database applications connect directly to databases using such connectivity tools, some problems may arise in regard to the access. If a malicious user is

J. Filipe et al. (Eds.): DATA 2017, CCIS 814, pp. 215–236, 2018.
https://doi.org/10.1007/978-3-319-94809-6_11

able to interact with the application that connects directly to the database, he can potentially obtain the database credentials and use them to connect to the database and disclose the data without restrictions. This can occur because applications may have to access a database using different accounts, and so it is easier to just hard-code the credentials into them or in configuration files, meaning that users of those applications do not need to know them. Additional security mechanism can be used to overcome this issue, such as securing the database access using virtual private networks (VPNs) which require another set of credentials to connect to.

A VPN can be used to access services in a wide (many services) or narrow (only one or very few services) approach. However, a wide VPN approach to access many services in a company has limited security, since anyone with access to one service may attempt to connect to another. Similarly, a narrow VPN approach does not protect against internal attacks that can bypass the VPN altogether. By compromising the VPN server, the database and all other services become much more exposed. Additionally, just looking at the client application source code or configuration files can often reveal the database credentials to attackers, granting them direct access to the database.

Using an authentication service on top of the database service is another possible solution to this problem, e.g. web services, but the interface of such services often do not support database tools such as JDBC, meaning that those tools must be dropped. Therefore, frameworks like Hibernate that rely on them cannot be used. Furthermore, simply removing the credentials from the application code is difficult since there may be many of them depending on the permissions and operations required to be done on the database, so providing the user of the application with all the credentials is not an option in most cases either.

The reason behind it is that, when a complex credential policy is in place, i.e. policies regarding password complexity and length [3], users would be prone to write the credentials down, making them easier to be disclosed. Various results also show that developers are aware of the security issues regarding passwords [4]. One might consider the usage of one-time passwords to secure the credentials, which also assumes the user has some sort of device to provide the password. However, database management systems (DBMS) do not usually support these types of authentication natively. Implementing such methods manually requires the exclusive use of stored procedures and post-connection validation, which becomes hard to manage in complex scenarios, or triggers to be added to each query, which decays the database performance.

This paper is an extension of the paper [5], where a solution called Secure Proxied Database Connectivity (SPDC) was presented which aims: to provide a mechanism that separates the information needed to connect to the database between a proxy server and an authentication server running within the database server; allow standard database connectivity tools, such as JDBC or ADO.NET, to be used; and take the database credentials out of the client application, so that a compromised client application does not disclose them. This solution requires a malicious user to compromise both servers to be able to establish a connection to the database and access its data. This extension includes an additional discussion section, another assessment test to verify the performance impact of the proxy server and the original arguments and methods presented in the previous paper.

This proposal emerged from the work done in [6–8], where a distributed access control framework allows the clients to connect to a database through runtime generated access control mechanisms. There, the generated access control mechanisms provide interfaces that wrap the JDBC connection, exposing only the methods that manipulate the data that the user has permission to perform them. A connection to a server side application is also established to configure the runtime generation of the access mechanisms, based on the security policy that applied to that user. While this eases the development effort to write a database application by giving developers interfaces tailored to the permissions given to the application, it lacks security in term of access to the database itself. SPDC was designed to enhance the access security to the database, while keeping support for the convenience of database connectivity tools, such as JDBC.

The paper is divided as follows: Sect. 2 presents the related work, Sect. 3 presents the core concepts of SPDC, Sect. 4 presents a proof-of-concept implementing SPDC, on Sect. 5 a performance assessment regarding network traffic is made, Sect. 6 finalizes with a brief discussion of the presented contents and Sect. 7 concludes the paper.

2 Related Work

To be best of our knowledge, there is not much work done to secure the JDBC or ADO.NET driver type protocols. A possible explanation is that most drivers used with these tools already support SSL/TLS [9] connections to DBMS using digital certificates. While this normally protects the credentials from being obtained from the network by eavesdropping it or through some other means, if the malicious users have access to the client application it may be possible to steal the credentials from there. Then directly accessing the database machine or connecting to it from within the organization's network and using the stolen credentials will grant access to those users.

While there is not much work done to secure the driver protocols, the SSL/TLS protocol on the other hand has seen some work to try to improve its security. In [10, 11], a session aware user authentication is introduced and expanded to thwart Man-In-The-Middle (MITM) attacks. Nevertheless, an internal attack on the database service to establish a direct connection is enough to allow a malicious user to access the data if the credentials are obtainable from the client application. In [12] a methodology is proposed to assist developers and database designers to design secure databases that follow the organization's guidelines for access control. It is applied and verified at the organizational and application development levels to ensure the satisfaction of the security requirements. The organizational level takes place before the application is developed and includes the organization's own security policies that are defined in security patterns. The application development level sees the data related security requirements defined into the data model, which is guided by the security patterns defined in the organizational level. While this approach helps protecting data from possible security breaches by implementing the security requirements in a methodical way, it does not protect the database from being read from or even from being modified directly if the credentials are stolen from the client application, depending on the client application permissions.

Another solution to this problem is to connect to the database through a web service instead of a direct connection to the database, such as a login service. While this has several advantages, such as only the server can establish connections to the database and the operations performed by the client applications can be more easily controlled, the clients must use the web service interface to access the database. The issue with this approach is that in most cases the interface provided differs from web service to web service. This means that the developers must master them, whereas tools such as JDBC and ADO.NET provide a well-known and well-established interface to access any supported database. The work presented in [13] leverages the importance of this problem, stating the need to create a unified interface for cloud data stores that have emerged recently. Other examples of this approach include the API to access Twitter, Facebook and other social networking services. However, in these cases, the issues are used as features to prevent developers from knowing the data schema, a situation that does not apply to applications built by an organization to access its own data. Furthermore, compromising the login service may be enough to access the database, since the malicious user knows the database endpoint and the database credentials are still configured in the login service application.

Web services can almost be seen as proxy servers that users use to access to manipulate the data stored in a database according to their permissions, albeit with their own interface. Proxy servers are used throughout the literature in some way to provide better security to access resources, be it a database, the Internet, etc. The work presented in [14, 15] show how proxy servers can be beneficial in both providing security and additional services. We restate that use of proxy servers in this paper aims to provide increased security by removing the need for client applications to possess real credentials to the database, while allowing them to use database connectivity tools transparently. Furthermore, a malicious user should only get access to the data in the database by successfully compromising both the proxy server and the authentication server, where other solutions usually only require one service to be compromised.

High-Availability JDBC, or HA-JDBC, [16] is an existing JDBC proxy project that has been implemented, is readily available and provides many features on top of what JDBC currently supports. However, it only focuses on being light-weight, transparent, and providing fault tolerant clustering capabilities to the underlying JDBC driver. Therefore, it still requires the client application to use the database credentials. In this light, it differs from the work done in this paper in which the client authenticates with the proxy itself using a server generated token and never uses the database credentials directly.

Other methods of authentication with databases have been proposed, such as using biometric data [17], digital certificates [18], third-party based authentication [19], etc. In the case of digital certificates, the application would have to store it within itself, meaning that it could be stolen just like we argue with username/password credential pairs. Additionally, SQL Server is able to use Microsoft Windows Integrated Security [20] to use the operating system's user account as the credentials used for authentication. While it does prevent the client applications from having credentials hard-coded, it now matters which account the user is logged into when using the application. Furthermore, normally an application only has one set of credentials to access a database. User biometric data suffers from the same issue and is not necessarily a safer or a more convenient option over the alternatives [21], since it can be possible to use photos for face recognition so legitimate users are always authenticated or the opposite happens, to prevent photos and masks from being able to authenticate a user then a number of legitimate users will be not authenticated, decreasing the usefulness of the system.

Table 1. State of the art summary in relation to SPDC features [5].

Solution	Cred. on the client	Nodes to breach	API	Other notes
Basic driver	Yes	Client	Standard	N/A
Oppliger et al.	Yes	Client	Standard	MITM mitigation
Abramov et al.	Yes	Client	Standard	Security guidelines
Webservice	No	Webservice	Custom	Access proxy
HA-JDBC	Yes	Client	Standard	Access proxy
Other auth methods	Yes	Client	Standard	May not be supported
VPN	Yes	Client	Standard	Internal access not prevented
SPDC	No	Auth./Proxy	Standard	Architecturally costly

A summary of the information presented on this section can be seen in Table 1, showing if the database credentials are on the client application, the nodes required to breach to get the credentials and the API provided to the client to access the database.

3 Solution Concept

In this section, the core concepts behind SPDC are presented. The objective is to offer a new approach to secure the access to a database while allowing standard tools, such as JDBC, to be used through a proxy server. Furthermore, this method is shown to protect the database from being accessed by a malicious user even if the information stored in the proxy server or the authentication server are disclosed individually.

To reiterate the issue SPDC is trying to solve, solutions like VPNs do protect access to services, such as databases, while still allowing client applications to use standard APIs such as JDBC to connect to them. However, as stated, VPN solutions can be used to access any number of services within a company. If a database service is provided within a VPN along with other services, then it becomes less secure, since users that can access one of the other services may attempt to connect to the database. If another login layer is used on top of the DBMS then other mechanisms must be implemented to support the usage of standard database connectivity tools, such as JDBC. In the case the database service is served in its own VPN, and similarly to the login layer solution just described, internal attacks can still leave the DBMS vulnerable to malicious users that have obtained the hard-coded credentials from the client applications. It only takes the VPN server to be compromised to leave a DBMS much more exposed, and if just by looking at a client application source code the database credentials can be obtained, then the data becomes accessible.

3.1 Conceptual Architecture

The SPDC approach was designed to protect the database by using a proxy server to communicate with the database, as shown in Fig. 1. The intent is to have the client connect to a proxy server, authenticating with it, and then have the proxy server connect to the database in place of the client using credentials it has stored, proceeding to relay the communication between them. The proxy server is then also intended to serve as a credentials provider, using the client application real credentials when establishing the connection to the database. These credentials are also encrypted to prevent them from being disclosed in plain text in the event the proxy server is compromised. This is where the authentication server mentioned comes into play.

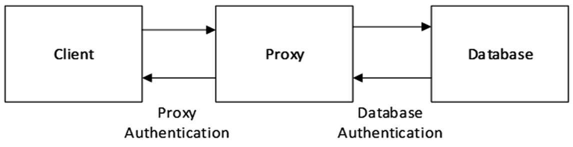


Fig. 1. Conceptual Client to Database connection [5].

It is stated that the presented solution protects a database even when the information in the proxy server or the authentication server is compromised. The only exception is if the DBMS itself is compromised in an internal attack and access is granted without proper authentication. To achieve this goal, both the proxy server and the authentication server were given an asymmetric key pair and the information required to access the DBMS was divided between them as follows:

- The **proxy server** is in possession of the user credentials, encrypted using a symmetric key *C* unique for each user. It does not store the symmetric key *C*.
- The **authentication server** is in possession of the symmetric key *C* for each user and can open endpoints to the DBMS for the proxy server to connect through. It does not store the credentials to the database.
- The **client application** possesses no relevant information for authentication with the database. Additionally, the user must know the credentials to connect to the authentication server. It does possess dummy credentials to a database.

Table 2 shows a summary of the information presented thus far. It is worth mentioning that the dummy credentials stored in the client application must have connection permissions to a database. This comes from a shortcoming in the database connectivity tools used. In the case of JDBC, a connection object can only be instantiated if a valid connection to a DBMS is successfully made, and a connection object is required for SPDC to establish the connection as intended. Nevertheless, an identical separate DBMS with no data or an account stripped of permissions may be

Table 2. Stored sensitive data per protocol participant [5].

Client application	Proxy server	Authentication server
Dummy database credentials	Encrypted user database credentials	Open database connection endpoints
		Symmetric keys C

used for this purpose, making the credentials stored in the client application effectively useless to access the data.

In this scenario, the user must first authenticate with the authentication server to obtain the server generated token and present it to the proxy server. The proxy server uses this token to authenticate the user and establish a valid connection to the DBMS endpoint.

3.2 Deployment Assumptions

Let us consider what happens when each party in this exchange is compromised and what other security technologies are assumed to be deployed. During the SPDC design, there were a few security assumptions that were made. Thus, they are important to be considered when creating security attack scenarios.

The initial assumption made was that the communication between all parties is protected to prevent the data from circulating in clear text within the network, either using TLS or some other mechanism to do so. Otherwise, credentials could be read from the network, invalidating any security enhancing properties of SPDC.

The second assumption was that the access to the network where the database and authentication server are deployed is restricted, e.g. by a firewall, allowing access to the database to be controlled and monitored. Furthermore, this would enable the authentication server to open endpoints to access the database only when requested.

The third assumption is that the credentials used to authenticate with the authentication server to obtain the token T are not be stored anywhere in the client. If the user cannot be trusted with a password, a solution based on one-time passwords, for example, should be used. These credentials would allow a malicious user to impersonate a legitimate one, and while the connection still had to be made through the SPDC system, the malicious user would be able to access, and possibly modify, part of the data.

Finally, the proxy server is assumed to be trusted by all the parties to not eavesdrop on data and to keep the database credentials protected. If this is not a possible scenario, the entity in charge of the database server should provide a proxy server on a distinct network with the same security requirements to access the database. It is also assumed that the asymmetric keys of the proxy and authentication servers are heavily secured and cannot be obtained easily by breaching the applications running on them, as it is expected of any server storing private keys.

3.3 Attack Scenarios

In this section, different attack scenarios that target distinct parts of the proposed solution are detailed and discussed.

Following the information shown in Table 2, if the **proxy server** is breached, then a malicious user would be in possession of the following data:

- The encrypted user credentials, which are a known function of the symmetric key *C* with the user username and password (or other form of authentication credentials). Without the symmetric key *C* to decrypt them only a brute force attack is available.

However, the following data is not in the malicious user possession:

- An open database endpoint to connect to, which can be random for every access attempt. This prevents the malicious user from attempting online attacks, however any open endpoints may be detectable using a port scan tool. Opening the endpoints exclusively for the proxy server would be a solution for this issue.
- The user symmetric key *C*, which is different for every user and that only the authentication server stores and provides inside a token, called the *T* token, given to the client application upon authentication which contains information such as the client application id, target database, the endpoint, expiration date, etc. This prevents the malicious user from decrypting the credentials if disclosed from the proxy server.

Attempting to impersonate the proxy server is impossible without stealing the proxy server asymmetric private key, since token *T* passed by the legitimate user is encrypted using the public key. Without it, the token *T* cannot be decrypted to retrieve the symmetric key *C* and the connection endpoint. Therefore, the connection to the database is impossible to be established, which also alerts the legitimate user that an attack could have occurred.

If the **authentication server** is compromised, then a malicious user would be in an inverted situation: he would possess the list of open database endpoints to connect to and the user symmetric keys *C*. However, without the encrypted credentials stored on the proxy server they are useless as he wouldn't be able to authenticate with the database itself.

Impersonating an authentication server is also useless to the malicious user since the legitimate users do not provide the database credentials for authentication. Since the users do not have to know the database credentials, these should be pseudo-randomly generated for each one of them.

Finally, if the **client application** is compromised, the malicious user is not capable of obtaining any relevant information to authenticate with the DBMS, since the credentials were pushed to the proxy server. Note again that the credentials used to obtain the token *T* are not meant to be stored on the client application, but to be known by the user of said application or using another authentication scheme such as one-time passwords.

3.4 Database Connection Proxying

In this section, the proxying procedure to secure access to databases while allowing the usage of standard database connectivity tools is shown. As stated, this is achieved by separating the authentication material between the authentication and a proxy server, which relays the connection to the database. Unfortunately, most database connectivity tools do not allow the usage of proxy servers or custom sockets to communicate with the DBMS, requiring the usage of alternatives such as reflection mechanisms to achieve it.

To setup the proxy server on the client application, it must first create a connection object to communicate with the database, such as a JDBC connection object, and then modify it internally so that it connects communicates with the proxy server instead. Furthermore, the proxy server is required to relay the communication between the client application and the DBMS connections. This setup allows the client application to interact with a database directly using a database connectivity tool while keeping the real database credentials unknown to the user.

In short, setting an existing generic communication socket to the proxy server to be used by a database connectivity tool connection will generally require the following steps:

1. Create a generic communication channel to connect the client application and the proxy server.
2. Obtain a token T after authenticating with the authentication server and pass it on to the proxy server.
3. Have the proxy server connect to the DBMS using the information stored on the token T provided by the client and start relaying the data between them.
4. The client application creates a connection object from the database connectivity tool of choice, using a DBMS account without privileges if required.
5. The client application sets the socket to be used by the database connectivity tool to the pre-existing generic communication socket with the proxy server. The database connectivity tool can now be used as normal.

This is implemented in the proof-of-concept using reflection mechanisms, in which the SQL Server DBMS and JDBC are used to implement this solution. The authentication server must ensure that an endpoint is open so the proxy server can connect.

3.5 Deployment Considerations

Implementing the concept discussed so far allows for a functional system to be implemented, but it is not enough to make it secure given the assumptions made in Sect. 3.2. All connections made must also be done using secure protocols, such as TLS, and the database cannot be accessed using information stored on the client application, proxy server or authentication server individually.

This would force a malicious user to compromise both the proxy server and authentication server to be able to access the data, unless the database can be exploited using the unprivileged account or access is forced in some other way. For this reason, a separate DBMS instance can be used for the purpose of creating the connection object on the client application. SPDC also allows to set arbitrarily complex passwords on the database accounts, since the clients are not meant to know them.

Hence, SPDC has a few limitations that should be taken into consideration when deploying.

First, it does not protect the network communications, depending on existing protocols, such as TLS, to do so. Its purpose is just to protect access to a database without disrupting existing database access APIs.

Second, if a malicious user were to be able to compromise the proxy server with an online attack to a point where the decrypted token is leaked from memory, then the user credentials could be obtained, as well as an endpoint to access the database. For this reason, the proxy server software must be robust to prevent memory from leaking.

Furthermore, the proxy server must be trusted or provided by the database server for the simple reason that, due to the proxying procedure, the proxy server would be able to eavesdrop on the communication. Finally, if a user is allowed to have a weak password to authenticate with the authentication server, the client application was modified by an attacker, or the communication between the client application and the authentication server is not protected, then it would be possible to impersonate the user.

SPDC can be enhanced with other security mechanisms and protocols, such as one-time passwords, to address this last issue to some degree.

Finally, the solution has a whole has an increased cost in the architecture, since it requires a proxy and an authentication server, and should be considered against the security level required for a use case.

4 Proof of Concept

In this section, the SPDC implementation details and how the client can connect to the database with credentials that grant them no read/write permissions on the database is presented. For the proof-of-concept, Java was used as the programming language, SQLServer 2012 as the RDBMS with the Northwind sample database and JDBC to connect to it. The JDBC driver used was the SQL Server official release for JDBC version 4.

As mentioned, the previous work relied on data structures that communicated with the DBMS to execute operations and retrieve data. However, the database credentials were statically defined in the application's source code, allowing any user to retrieve those using mechanisms like reflection or by analyzing the source code. This was a big security vulnerability, since the user had access to the credentials used to connect to the database with the sensitive data. This allowed him to bypass the security components completely and connect directly to the DBMS outside the application. Furthermore, sensitive information like the credentials and data was transmitted through the network in clear text, so any eavesdropper could see it.

To implement SPDC using JDBC, the following structures/mechanisms were then implemented:

1. The token *T*, generated by the database side authentication service when the client application authenticates.

2. The JDBC connection object modification procedure to use a custom communication socket so that the client application can use it while having the communication relayed through the proxy server. This procedure is different for other database access drivers and tools.
3. The relaying mechanism between the database and the client application.
4. A mechanism to open endpoints to access the database on the database server side.

For the first point, a JSON (JavaScript Object Notation) object was used to encode the token. It is comprised of two main parts: the *token_data* and the *token_sig*. The *token_data* is another JSON object that contains the fields necessary for the proxy server to establish a database connection. In this proof-of-concept the *token_data* contains the following fields:

- **username:** The username of the user attempting to connect. This is an encrypted field.
- **database:** The name of the database being requested access. This is an encrypted field.
- **created:** The timestamp of the token creation.
- **expires:** A timestamp of the instant the token expires and can no longer be used.
- **nounce:** A random 32-bit integer.
- **endpoint:** The IP address and port to which the proxy must connect. This is an encrypted field.
- **C:** The symmetric key associated to the user, required to decrypt the credentials stored by the proxy server. This is an encrypted field.

All encrypted fields are encrypted using the proxy public key, shared with the authentication server *a-priori*. The symmetric key *C* is first generated when the user is given an account for the database. The associated credentials are then encrypted with it and sent to the proxy server securely. Since the proxy server must decrypt the credentials to establish a database connection, it is possible to update the symmetric key periodically by providing both the old and the new symmetric keys.

The second half of the token, the field *token_sig*, contains a signature of the hash of the *token_data* field, signed using the authentication server private key so that the proxy server can validate it.

Considering the second point presented, some problems were found when trying to implement it. As discussed in Sect. 3.4, the current JDBC API does not allow an existing socket to be used to connect to the database and instead requires a connection to be made directly. Using a connection string and successfully connecting to the database is the only option available to get the connection object instantiated, which is required to make requests to the database. The problem is that this connection object is then connected to a DBMS directly and not the proxy server.

It was determined that to create a connection object, the real credentials are not required, just the credentials to an account that has no permissions to execute queries, i.e. an unprivileged public account, provides the client with the necessary connection object. To have the connection object communicate with the DBMS through the proxy server, the client application can utilize reflection mechanisms. Using reflection, the internal socket and input/output streams can be altered to the generic communication channel streams used earlier to communicate with the proxy server. However, the usage of reflection mechanisms to achieve this has the downside of making the proof-of-concept work only for the Microsoft SQLServer driver. Other drivers may use other internal structures for the connection object, requiring a deep analysis of them to make them work with SPDC as it stands.

The connection class implementation of JDBC for SQL Server has several variables, but one is of particular interest: a variable named *tdsChannel*. This is the variable that contains the socket, named *channelSocket*, two identical input streams, named *inputStream* and *tcpInputStream*, and finally two identical output streams, named *outputStream* and *tcpOutputStream*. When it is said that they are identical, it is meant that they hold the same reference when the driver is in use. The input/output streams are the streams used by the socket to read and write data.

With this information, it is easy to understand how the JDBC connection object can be altered so that is communicates with the proxy instead of the database. All that is needed is to set the proxy socket and its input/output stream object references as the new values for the variables mentioned.

For the third point, the relaying between the client and the database performed by the proxy server is achieved using two worker threads, one for each communication direction, and the Apache Commons IO library for copying the data between the socket streams since it provides this feature. With this setup, the client can continue to use the original connection object that was created to communicate with the database without ever knowing the real account credentials.

For the final point, commands to the firewall can be used to open and close ports to access the database by the authentication server as required.

To summarize, Fig. 2 shows a diagram of the SPDC implementation, which is comprised of five main steps:

1. The client application instantiates the JDBC Connection object, using an unprivileged account on the database to do so.
2. The client application then connects to the authentication server using the credentials given by the user. The authentication server generates and sends to the client application a security token T with the necessary information for the proxy server to be able to connect to the database. An access endpoint is also opened.
3. The client application connects to the proxy server, presents the token T, waits for the token to be validated and executes the procedure to alter the JDBC Connection object as described. The proxy server decrypts the token T when it is received and validates the authentication server signature before replying to the application whether the token is valid or not.
4. Using the symmetric key C within the token T, the proxy decrypts the credentials stored for that user and establishes a connection to the database using the endpoint indicated in the token.
5. The proxy server relays the data between the client application and the database.

Tables 3 and 4 show the network protocol implemented in the proof-of-concept. Table 3 shows the network messages sent between the client application and the authentication server that runs on the database machine. It starts (1) with the client application presenting a username u and password p pair to the authentication server,

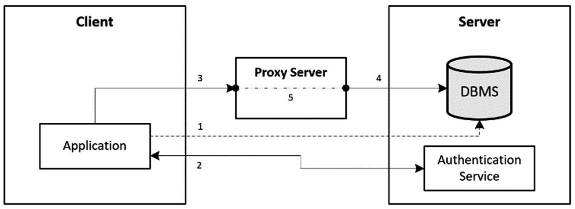


Fig. 2. Sequence implementation diagram of the solution [5].

Table 3. Protocol messages between the client application and database server running the authentication server [5].

#	Client application	Network	Authentication server
1	*u, p*	$\rightarrow$ (*u, p*)	
2	*u, p*	$\leftarrow$ (OK|NOK)	*u, p,* auth(*u, p*)
3	*u, p*	$\leftarrow$ (*T*)	*u, p,* *T* = genToken(*u*)
4	*u, p, T*		

Table 4. Protocol messages between the client application and proxy server [5].

#	Client application	Network	Proxy server
1	*T, O* = connectToDatabase()	$\rightarrow$ (*T*)	
2	*O*	$\leftarrow$ (OK|NOK)	*T,* validateToken(*T*)
3	*O,* useProxySocket(*O*)		*T,* connectToDatabase(*T*)
4	*O,* applicationCode(*O*)		relay()

which authenticates (2) using its own password database. The authentication result is sent back to the client application with a successful message ("OK") or not successful ("NOK"). If the authentication is successful, then the authentication server generates a token T and (3) sends it back to the client application. If the authentication is not successful, the communication is terminated and additional security measures may be taken to prevent online password attacks. This completes the communication between the client application and the authentication server (4).

Table 4 shows the network messages between the client application and the proxy server. It starts (1) with the client application presenting the token T it obtained from the authentication server beforehand and connecting to the database using the public account. The proxy server decodes and validates the received token *T* (2), replying to the user with a validation successful message ("OK") or not successful ("NOK"). Again, If the token *T* is valid, then the proxy server connects to the database using the information stored within the token *T* and the credentials stored in its own credentials database (3). This process also includes retrieving the stored encrypted credentials and decrypting them using the symmetric key *C* in the token *T*. At the same time, the client application sets the connection object it has with the database to use the network socket connected to the proxy server. If the authentication is not successful, the communication is terminated and additional security measures may be taken to prevent online attacks to crack the authentication server private key. Finally, the proxy server begins relaying the communication between the client application and the database while the client application executes the application code (4).

All unnecessary variables and attributes are removed from memory when possible to minimize the chance that they may be leaked. This is most evident with the token *T*, which the client application only stores until it is sent to the proxy server. Furthermore, the proxy server only keeps the token *T* in memory until the database connection is established.

Other more complete authentication protocols can be used to authenticate the user, such as Kerberos [22] or RADIUS [23] and even implementing the GSSAPI [24] to support a wide variety of protocols, since SPDC was design to work independently of the method used to authenticate users. SPDC is only concerned with the proxy server obtaining the information required to establish a database connection and relay the communication to the client, which is why this proof-of-concept uses a basic authentication protocol.

Table 5. Testing machines specification [5].

Machine	A	B
OS	Windows 10 Home	Windows 7 Ultimate
Architecture	x86_64	x86_64
Motherboard	LENOVO Lancer 5A2 (U3E1)	Gigabyte H87-HD3
CPU	Intel Core i7 4510U @2.00 GHz	Intel Core i5 4670 K @3.40 GHz
Memory	8.00 GB DDR3 @797 MHz	8.00 GB DDR3 @665 MHz
Hard Drive	465 GB SSHD-8 GB	1863 GB SATA
RDBMS	Microsoft SQL Server 2012	–
Other Programs	Netbeans IDE, Wireshark	Netbeans IDE, Wireshark

5 Performance Assessment

In this section, the performance assessment made to test the solution is presented. This performance assessment includes a comparison between the network traffic generated by a VPN, the solution proposed in this paper and the direct, unprotected JDBC. Included in this performance assessment is also a load test where the number of clients was increased and the average time per query was measured with a single proxy server and two proxy servers.

5.1 Network Traffic Assessment

This section will detail the network traffic tests performed between the relay method used by SPDC, the direct connection method through a VPN and the direct JDBC connection. Each one was carried out by establishing the initial connection and then issuing an aggregation query to the SQLServer Northwind sample database that calculates the total amount of money per order, sorting by the total. The performance in terms of delay is not included because everything is done during connection time and will be discussed in more detail in Sect. 6.

The machines used to carry these tests are specified on Table 5 and the network communication was captured using Wireshark v2.2.2.

The direct approach using a VPN is trivial and was conducted by instantiating a JDBC connection object and then executing an aggregation query over Northwind's Orders table which contains 1000 rows.

A third-party VPN server, supported by OpenVPN, was used to establish the VPN connection. The network traffic was captured by the client application to measure the amount of data sent through the VPN. The SPDC solution uses TLS to secure the connections described in this paper and was deployed on two machines: machine A and machine B. The test cases using a VPN and SPDC are illustrated in Fig. 3, which also shows which role each machine had. The unprotected, direct JDBC connection was made from machine B to A.

To reiterate, after the client application authenticates with the authentication server, it waits for the authentication server to reply with the token T as described on Sect. 3. Additionally, the authentication server opens an endpoint on the firewall, so the proxy server can connect using the information provided in the token *T*. The client application (machine A) then connects to the proxy server (machine B), and sends the token it received from the authentication server. At the same time, it connects to a DBMS instance using the unprivileged account to instantiate a JDBC connection object. Once the proxy server acknowledges that the token is valid, the client application modifies the JDBC connection object so the socket connected to the proxy server is used instead. At the same time, the proxy server connects to the database using the user credentials that it decrypted with the symmetric key C from the token, and begins relaying the communication between the client application and the database.

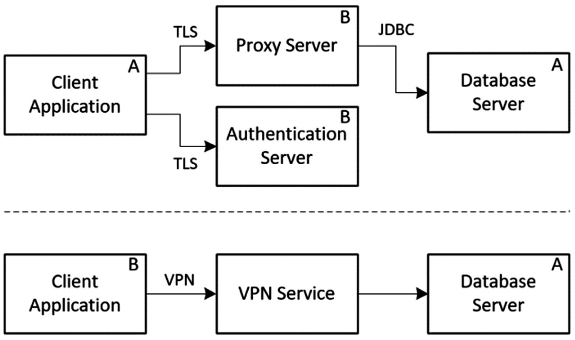


Fig. 3. Test deployments, the top case using SPDC and the bottom using a VPN [5].

The query execution made by the client is the same as in the VPN approach, since it now possesses a JDBC connection object with permission to access the data on the database. From this point on, SPDC adds no further traffic overhead to the connection, meaning that it no longer has any impact after the connection is established.

Table 6. Traffic overhead results [5].

Solution	#Packets	Bytes transmitted
Connection: **VPN**	177	31409
Connection: **SPDC**	48	19303
Connection: **JDBC**	13	2853
100 Queries: **VPN**	2356	1480972
100 Queries: **SPDC**	1649	1316834
100 Queries: **JDBC**	1507	1295578

On Table 6 a summary of the results obtained when capturing the traffic generated by each solution can be seen. On every solution, the relevant network traffic was captured by the client application. To ensure no unwanted traffic was captured on the VPN solution, the VPN server was configured to only allow connections made to the database server through.

It is possible to see that overall, SPDC using TLS to secure the connections is better than a VPN based solution in regards to network traffic. During connection, SPDC used around 73% less packets than the VPN counterpart (just 48 packets compared to the 177 used by the VPN). Furthermore, SPDC transmitted about 39% less data (just 19 KB compared to the 31 KB used by the VPN). This is expected since VPN must also transmit information regarding routes and other network related information so the client can setup the VPN successfully. Furthermore, both solutions add a lot of overhead when compared to an unprotected connection, which only needed 13 packets to transmit a little under 3 KB of data. This can be explained due to the authentication protocols used in both solutions and it is the trade-off to get better security when accessing the database.

Considering the applicational data transmission with the database queries, the SPDC transmitted 30% less packets (1649 packets when the VPN solution needed 2356). This can be explained due to the fact that SPDC, after the connection is established, has no other traffic overhead. Thus, the overhead in the SPDC solution is added just by the TLS protocol itself. The same reason can be applied to explain the data transmitted, since SPDC was able to serve 100 database queries while transmitting 11% less data than the VPN solution (about 1.26 MB compared to the 1.41 MB needed by the VPN). Unsurprisingly, the unprotected connection to the database required less packets and bytes transmitted for the 100 queries, around 63% of the packets needed for the VPN solution and 91% for the SPDC solution. The number of bytes transmitted is even closer, requiring 87% of the bytes transmitted in the VPN solution and 98% for the SPDC solution. This shows that the overhead of the SPDC solution for the actual data transmission is almost non-existent, having similar performance to the unprotected JDBC connection.

It is important to note that it is possible to have the proxy server establish a VPN connection to the database server before connecting to the database. In this case, the network traffic overhead is expected to be the same as if only a VPN was being used, since SPDC adds no overhead once the connection is established.

5.2 Client Load Assessment

With the purpose of testing the performance degradation of the proxy server with the increasing number of connections, a client load assessment test was also carried out.

This test aimed to time the execution of the query used for the network tests with an increasing number of concurrent clients doing so. Moreover, this assessment was conducted with a single proxy server and two proxy servers.

In the single proxy server setup, nothing was changed from the network tests for SPDC, except the machine hosting the clients spawned more client threads connecting to the database via the proxy. In the double proxy setup, a third machine was used which functioned only as an additional proxy server. The clients would then each authenticate with the authentication server and connect to each proxy server using a round robin load balancer.

Figure 4 shows the results of this assessment, ranging from 1 to 32 concurrent clients accessing the database and the average time per query.

One of the first conclusions that can be taken from the graph is that apparently, the usage of one or two proxy servers did not change the performance of the system.

There are two possible explanations for this outcome. The first reason is that the client machine might not have been powerful enough to handle all the client threads.

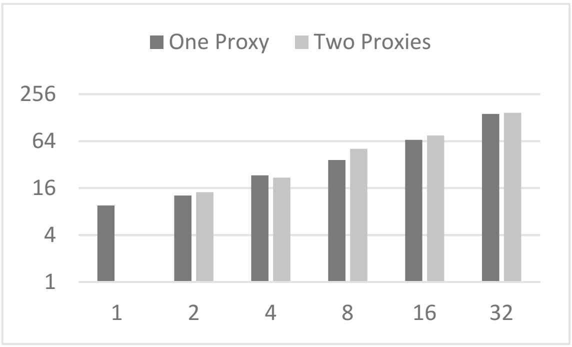


Fig. 4. Time in ms per query by increasing number of concurrent clients.

The second reason is that the database machine is not powerful enough to handle all the incoming queries. Regarding the first reason, a test was conducted where two different machines were used to spawn clients and it showed similar results to the ones display on Fig. 4, meaning that the bottleneck was not on the client machine used. Therefore, it was concluded that the database server itself could not handle the number of queries being made without the degradation in performance obtained.

However, this also means that the proxy server was not the bottleneck. Thus, when deploying this solution, the database server is the component that should be given the most attention in terms of computational resources.

While this assessment showed that the proxy server is not as much of a bottleneck as the database server can be, once the database server has enough power to handle all the incoming requests the proxy server might be the next point of resource contention. Unfortunately, it is difficult to test this without a powerful machine hosting the database, as each connection to the database much keep transmitting data to keep the proxy server busy.

6 Discussion

So far, the methodology to enhance the security of database access was introduced, detailed and evaluated.

The methodology consists of taking the database credentials, encrypting them using a symmetric key, known only to the authentication server, and storing them on the proxy server. Thus, breaching the authentication server discloses only the symmetric keys to a malicious user, while breaching the proxy server discloses only the encrypted credentials.

While this is true, the client application needs to authenticate with the authentication server so it knows which symmetric key to send to the proxy server. It is imperative that the credentials used by the client application for this purpose are not stored anywhere, and this perhaps the weakest link in the methodology. However, this is still better than storing the database credentials on the application itself. First, the user of the client application needs only to memorize one set of credentials instead of possibly many, which is one of the reasons the credentials are hard coded in the first place. Second, the malicious user still must use the system to access the database, as the credentials in the client application are used only for the authentication server and not the database. This prevents internal attacks using database credentials from bypassing the entire system, and as such monitoring tools can still be used which may detect the access as suspicious.

A second issue that was identified was the moment the proxy server establishes the connection to the database for a client application. In that moment, it must decrypt the database credentials and use them. It is important to ensure that the proxy server is not breached and memory leaked, which may contain the decrypted credentials. It is easier to do since the proxy server can run in an environment that is controlled and heavily secured, where the client application may run in a (semi-)public location.

Another weak link is not related to the database access itself, but instead the data that is transmitted to and from the client application and the database. The proxy server must relay the data between them, meaning that if the proxy server application is breached, it could be possible to eavesdrop and even modify the data.

It is clear that this methodology is not completely secure is every regard, and that was not the intention either. The intent is for this approach to serve as a base for more secure access to databases and perhaps other services as well. In fact, there is nothing in the methodology that restricts it to access to databases. So long as the proxy server can establish a connection to a service and the client application can create one as well without having to authenticate, then the proxy server can be used for authentication and relay the access between the service and the client application. Of course, this may arise other security issues, depending on the service and if the communication was already secure to begin with. Database access was targeted because credentials are usually hard-coded and communication is done in clear text, which may not apply to every scenario.

Finally, there were a few choices made in terms of the proof of concept and performance assessment. Mainly, the performance assessment was done by comparing the number of packets and bytes transmitted between the SPDC and a VPN approach. This was done because approaches other than VPN destroy the JDBC abstraction by using different APIs, which was meant to be avoided with SPDC. Additionally, no measure of time was conducted because a VPN must be setup manually before the connection to the database is opened, which would skew the measurements. Furthermore, connection time is a non-issue, since a connection can be made and maintained, so many queries can be done without having to reconnect. Therefore, the number of packets and bytes transmitted were found to be more useful than connection time.

One last possible concern with SPDC and the use of a proxy server that could be raised regarding scaling, since the proxy server must relay the communication between the client and the DBMS. However, there is no issue in having several of these proxy servers to process the client's requests to connect to the database since they are inherently stateless. They establish a connection given a token, and once the connection terminates a new connection can only be made by presenting another token. Thus, it is possible to have a load balancer in front of N proxy servers to help scale the solution. This was proven to work on the second performance assessment made, where both a single and two proxy servers were used to determine the performance degradation the proxy server would impose in the system as the number of clients connected to the database increased.

However, it was found that the bottleneck on the system was the database server and not the proxy server, since the clients needed to keep issuing queries to keep transmitting data through the proxy servers. While this provides some insight into the fact that the proxy server is not limiting the system too much, the testing scenario was not carried out with enough computational resources to find the impact that the proxy server has on the system. Since the throughput of the database would increase with more computational resources, it would also place an increased load on the proxy server. It is clear that an adequate testbed for SPDC, when it comes to the proxy server impact on performance, is required and is something that should be carried out when possible.

Nevertheless, the main application scenario for SPDC is to protect the direct database access made by applications within an organization, where the application can run on (semi-)public locations. As such, the number of connections to the database is better known than in other scenarios, such as a webserver which cannot know how many users will attempt to connect. Therefore, and given the results obtained in these assessments, the performance of SPDC seems to be quite satisfactory.

7 Conclusion

This paper presented and further detailed a mechanism to secure access to databases using a proxy server while allowing client applications to access data in them while using standard database connectivity tools. This approach was called SPDC, previously presented in [5], and it splits the information needed to access the database between the proxy server and the authentication server, as well as removing the hard-coded database credentials from client applications to achieve the aforementioned goal.

This way, a malicious user with access to the client application is not able to obtain database credentials just by looking at the source code. By having the proxy server connect to the database using the credentials associated with the client and relaying the communication between the client and the database, it is possible for the client to connect to the database using database connectivity tools without knowing the database credentials. Since neither the proxy server or the authentication server possess all the information needed to access the database, a malicious user must compromise both servers to be able to gain access to the data.

However, several aspects of this solution must be taken into consideration. The cost of the architecture is greater than that of a VPN solution, for example, since it requires a proxy server and an authentication server, in contrast to only one additional server in the case of a VPN. Furthermore, the proxy server must be carefully monitored to avoid eavesdropping, given the nature of proxy servers the data must be taken from one secure channel to another. This data should also not remain in memory or be cached, due to the possibility of a malicious user being able to access the memory once inside the proxy server. Additionally, and while it is not a very hard adaptation to make, JDBC drivers other than SQL Server's must have a procedure created to modify the communication sockets, so the objects created interact with the proxy server instead.

Furthermore, some possible attack scenarios were laid out and assessed to determine the pieces of information that a malicious user would have access to. Assuming that the credentials to the authentication server are not hard-coded or written down, the information disclosed in each case was only enough to grant access to the database if both the proxy server and the authentication server were compromised.

Concerning performance, it was shown that connecting to the database using the proposed SPDC mechanism can be better than using a VPN solution in terms of network traffic overhead, both in terms of connection and applicational data transmission.

To determine whether the proxy server was a main source of overhead, a performance assessment was also carried out. It showed that the main source of overhead was the database server, meaning that the proxy server does not require as many computational resources.

Finally, in terms of future work it would be interesting to address the eavesdropping issue on the proxy server. This could be achieved by adding a layer on top of the database that accepts connections and all data is encrypted using, for example, the authentication server password of the user. The implications of such alteration must be carefully studied and tested.

Furthermore, additional testing with a suitable testbed to determine the performance degradation of the proxy server with increasing loads would be interesting to conduct, so a proper scaling approach can be identified.

Acknowledgements. This work is funded by National Funds through FCT - Fundação para a Ciência e a Tecnologia under the project UID/EEA/50008/2013 and SFRH/BD/109911/2015.

References

1. Oracle JDBC Introduction (1997). http://docs.oracle.com/javase/tutorial/jdbc/overview/index.html. Accessed 3 Mar 2014
2. Bauer, C., King, G.: Hibernate in Action (2005)
3. Shay, R., Cranor, L.F., Komanduri, S., et al.: Designing password policies for strength and usability. ACM Trans. Inf. Syst. Secur. **18**, 1–34 (2016). https://doi.org/10.1145/2891411
4. Yang, X.L., Lo, D., Xia, X., et al.: What security questions do developers ask? a large-scale study of stack overflow posts. J. Comput. Sci. Technol. **31**, 910–924 (2016). https://doi.org/10.1007/s11390-016-1672-0
5. Regateiro, D.D., Pereira, Ó.M., Aguiar, R.L.: SPDC: secure proxied database connectivity. In: 6th Data - International Conference Data Science Technology Applications (2017)
6. Pereira, Ó.M., Regateiro, D.D., Aguiar, R.L.: Secure, dynamic and distributed access control stack for database applications. Int. J. Softw. Eng. Knowl. Eng. **25**, 1703–1708 (2015). https://doi.org/10.1142/S0218194015710035
7. Regateiro, D.D., Pereira, Ó.M., Aguiar, R.L.: A secure, distributed and dynamic RBAC for relational applications. University of Aveiro (2014)
8. Pereira, O.M., Regateiro, D.D., Aguiar, R.L.: Role-based access control mechanisms. In: 2014 IEEE Symposium Computers and Communications, pp. 1–7. IEEE, Vancouver (2014)
9. IETF (2008) RFC 5246: The Transport Layer Security (TLS) Protocol - Version 1.2. http://tools.ietf.org/html/rfc5246
10. Oppliger, R., Hauser, R., Basin, D.: SSL/TLS session-aware user authentication - Or how to effectively thwart the man-in-the-middle. Comput. Commun. **29**, 2238–2246 (2006). https://doi.org/10.1016/j.comcom.2006.03.004
11. Oppliger, R., Hauser, R., Basin, D.: SSL/TLS session-aware user authentication revisited. Comput. Secur. **27**, 64–70 (2008). https://doi.org/10.1016/j.cose.2008.04.005
12. Abramov, J., Anson, O., Dahan, M., et al.: A methodology for integrating access control policies within database development. Comput. Secur. **31**, 299–314 (2012). https://doi.org/10.1016/j.cose.2012.01.004
13. Gessert, F., Friedrich, S., Wingerath, W., et al.: Towards a scalable and unified REST API for cloud data stores. Lect Notes Informatics (LNI), Proc - Ser Gesellschaft fur Inform P-232, pp. 723–734 (2014)
14. Zarnett, J., Tripunitara, M., Lam, P.: Role-based access control (RBAC) in Java via proxy objects using annotations. In: Proceedings of 15th ACM Symposium Access Control Model Technology-SACMAT 2010, p. 79 (2010). https://doi.org/10.1145/1809842.1809858
15. Naylor, D., Schomp, K., Varvello, M., et al.: Multi-context TLS (mcTLS). ACM SIGCOMM Comput. Commun. Rev. **45**, 199–212 (2015). https://doi.org/10.1145/2829988.2787482
16. Ferraro, P.: HA-JDBC: High-Availability JDBC. https://ha-jdbc.github.io. Accessed 13 Sep 2016

17. Villager, C., Dittmann, J.: Biometrics for user authentication. Encyclopedia of Multimedia, pp. 48–55. Springer, Boston (2008)
18. de Lavarene, J.: SSL With Oracle JDBC Thin Driver (2010)
19. Oracle Authentication Using Third-Party Services. https://docs.oracle.com/cd/B19306_01/network.102/b14266/authmeth.htm#i1009853. Accessed 13 Aug 2016
20. Microsoft SQL Server Security Modes. https://msdn.microsoft.com/en-us/library/aa266913(v=vs.60).aspx. Accessed 13 Sep 2016
21. Zimmerman, M.: Biometrics and User Authentication (2003)
22. Neuman, C.B., Ts'o, T.: Kerberos: an authentication service for computer networks. In: IEEE Communications Magazine, pp. 33–38 (1994)
23. IETF (2000) RFC 2865: Remote Authentication Dial In User Service (RADIUS). https://tools.ietf.org/html/rfc2865
24. IETF (2000) RFC 2743: Generic Security Service Application Program Interface Version 2, Update 1. https://tools.ietf.org/html/rfc2743

ChronoGraph: A Versioned TinkerPop Graph Database

Martin Haeusler(✉), Thomas Trojer, Johannes Kessler, Matthias Farwick, Emmanuel Nowakowski, and Ruth Breu

Institute for Computer Science, Technikerstraße 21a, Innsbruck, Austria
{martin.haeusler,thomas.trojer,johannes.kessler,matthias.farwick, emmanuel.nowakowski,ruth.breu}@uibk.ac.at

Abstract. Database content versioning is an established concept in modern SQL databases, which also became part of the SQL standard in 2011. It is used in business applications to support features such as traceability of changes, auditing, historical data analysis and trend analysis. However, versioning capabilities have barely been considered outside of the relational context so far. In particular in the emerging graph technologies, these aspects are being neglected by database vendors. This paper presents ChronoGraph (This work was partially funded by the research project “txtureSA” (FWF-Project P 29022).), the first full-featured TinkerPop-compliant graph database that provides support for transparent system-time content versioning and analysis. This paper offers two key contributions: We present the concepts and architecture of ChronoGraph as a new addition to the state of the art in graph databases, and we also provide our implementation as an open-source project. In order to demonstrate the feasibility of our proposed solution, we compare it with existing, non-versioned graph databases in a controlled experiment.

1 Introduction

Graph databases offer a powerful alternative to traditional relational databases, especially in cases where most information value is found in relationships between elements, rather than their properties. Particularly with the popular and commercially successful graph database Neo4j[1], the concept of graph databases has reached a wider audience and has inspired many other implementations, such as Titan DB[2] and Orient DB[3].

Much like SQL for relational databases, the property graph model [23] Apache TinkerPop[4] (alongside the traversal language Gremlin) is the de-facto standard interface for graph databases, allowing to exchange the actual database

[1] https://neo4j.com/.
[2] http://titan.thinkaurelius.com/.
[3] http://orientdb.com/.
[4] https://tinkerpop.apache.org/.

© Springer International Publishing AG, part of Springer Nature 2018
J. Filipe et al. (Eds.): DATA 2017, CCIS 814, pp. 237–260, 2018.
https://doi.org/10.1007/978-3-319-94809-6_12

implementation without altering the application. As these technologies mature over time, they are faced with new demands for features. Among them is the demand for system-time content versioning, primarily for the purpose of maintaining traceability of changes, providing extensive auditing capabilities, legal compliance or historical data analysis of the graph contents (e.g. trend analysis).

The concept of versioning database content is a well-known topic. Early work dates back to 1986 when Richard Snodgrass published his article *Temporal Databases* [28]. In the following years, several different approaches were discussed [2,5,14,17,19], with many of them focusing on a relational environment and SQL (e.g. Oracle Flashback [12], Temporal Tables in DB2 [25] or ImmortalDB for SQL Server [15,16]).

Versioning for graph databases is a more recent topic [3,29,30]. Castelltort et al. and Taentzer et al. have shown clearly that achieving full-featured graph versioning within a given non-versioned general purpose graph database is a challenging task. It often leads to a sharp increase of complexity in the graph structure and consequently causes issues regarding scalability, comprehensibility and performance of queries operating on the graph. Due to the different trade-offs in individual solutions, to the best of our knowledge there is no generally accepted algorithm for constructing a graph such that it can effectively support multiple versions at the same time.

With ChronoGraph[5] [10], the first versioned TinkerPop implementation, we propose a novel solution that retains the simplicity and ease of use of graph queries on a non-versioned graph by handling the versioning process in an entirely transparent way, utilizing a three-tier architecture. We are also going to showcase scenarios where versioning provides advantages in the execution of regular graph database features. As our system is already being used in practice, achieving a high performance alongside new versioning features is important. We performed a benchmark to evaluate the performance of our system, which produced very promising results.

The remainder of this paper is structured as follows: In Sect. 2 we present the individual requirements which we considered for our solution. Section 3 introduces the existing key technologies on which this solution is based. In Sect. 4 we focus on the solution architecture and selected details, which are discussed in depth and compared with related work in Sect. 5. Section 6 presents an evaluation of our approach through controlled experiments. Finally, Sect. 7 outlines our future work and Sect. 8 concludes the paper with a summary.

2 Requirements

Based on our previous work with graph databases [6] and versioning systems [8], as well as by translating versioning features in SQL [12,15,16,25] to their corresponding graph counterparts, we synthesized the following key requirements for a versioned graph database:

[5] http://tinyurl.com/chronograph-github.

- **R1: Any Query on any Timestamp**
  In order to provide effective comparisons between individual graph versions, the essential underlying capability is to execute any given Gremlin query on any desired timestamp, without altering the query. In this way, comparisons between version a and b can invoke query q on a and b separately, and the query results can be compared directly.
- **R2: Efficient "Time Travel"**
  The ability to request the graph state at a given timestamp in the past is referred to as "Time Travel". This operation needs to be implemented in an efficient way in order to support requirement [**R1**]. Specifically, we demand that the query response time does not depend on the request timestamp. In other words, requests on timestamps far in the past should not perform inherently worse than requests on recent timestamps.
- **R3: History Analysis for Vertices & Edges**
  The ability to list the timestamps at which a given vertex or edge was changed is essential for analyzing the history of the element. At these timestamps, the analysis query in question can be repeated for comparison purposes.
- **R4: Listing Change Timestamps**
  A versioned graph database should be capable of listing all changed elements in a given time range and their corresponding change timestamps, e.g. for calculating deltas. This list is not restricted to any particular element, therefore this requirement is orthogonal to requirement [**R3**].

Given an implementation that meets these requirements, and the Gremlin query language, an application is able to effectively navigate in the graph as well as in the graph history.

3 Technical Background

In this section, we present the key technologies and concepts that form the foundation of our work. We first introduce our previous work, then we give an overview on the TinkerPop standard.

3.1 ChronoDB

In his paper *Efficient indexing for constraint and temporal databases* [22] from 1997, Sidhar Ramaswamy stated that the data versioning problem "is a generalization of two dimensional search which is known to be hard to solve". In our experience, the complexity of the problem further increases with the expressiveness of the chosen data model. For that reason, we decided to choose a simple base format and then transform the more sophisticated graph structures into that format (see Sect. 4.2). The format of choice here is a Key–Value Store that is enhanced with temporal information. Over the years, several different authors published a variety of approaches for versioning in this format [7,15,22]. Our

ChronoDB [8] project[6] is inspired by these ideas. This Temporal Key–Value Store operates on triples

$$Entry := \langle t, k, v \rangle$$

...where t is a Unix-style 64 bit timestamp, k is an arbitrary string that acts as the key, and v is the value associated with the key, which is a byte array of arbitrary length greater zero. In contrast to other formalizations (e.g. the time window approach proposed by Ramaswamy [22]) our approach does not involve *validity ranges* in terms of lower and upper timestamp bounds. Instead, in our case the validity range is *implicit*—any given entry is valid until a new entry with the same key is inserted that has a higher timestamp, "overriding" the previous entry. This can be visualized as a matrix, as shown in Fig. 1.

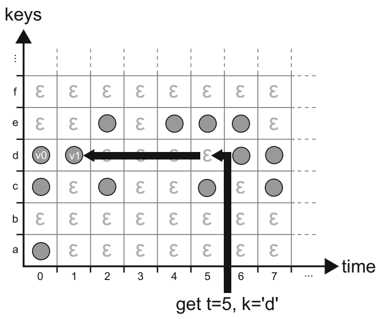


Fig. 1. Performing a *get* operation on a temporal data matrix [8].

This figure shows the process of retrieving the value for key d at timestamp 5. The algorithm first tries to find the entry at the exact location, and then backtracks and checks older timestamps. The first value encountered during this search is returned. A key that does not have a new entry from one timestamp to the next is assumed to have remained unchanged. Since entries, once written into the database, are considered immutable, copying their data is not necessary, thus maximizing sharing of unchanged data between versions. Furthermore, the fact that upper bounds of validity ranges are implicit in this structure, data that was once written to disk never needs to be modified again, allowing the implementation to be a true *append only* store. A modification or insertion both result in the execution of a *put* operation that inserts new entries. To ensure consistency of the history, new entries may only be added *after* the last entry on the time dimension, i.e. modification of the past is prohibited.

Our current implementation of ChronoDB relies on a B$^+$-Tree structure [24] for its primary index, which is a regular B-Tree where the leaf nodes in addition form a linked list for fast neighbour search. In this index, the key (which we refer to as *temporal key*) is a string consisting of the original user key, a separator,

[6] http://tinyurl.com/chronodb-github.

Table 1. Temporal key ordering by example [8].

Order	Temporal key	User key	Timestamp
0	`a@0123`	a	123
1	`a@0124`	a	124
2	`a@1000`	a	1000
3	`aa@0100`	aa	100
4	`b@0001`	b	1
5	`ba@0001`	ba	1

and the timestamp of insertion (left-padded with zeros to give equal length to all timestamps). The tree is sorted by the lexicographic ordering on the temporal key, resulting in the order displayed in Table 1. This ordering is critical for the implementation, because executing a temporal *get* operation as depicted in Fig. 1 is now equivalent to finding the temporal key where the user key is identical to the given one, and the timestamp is either equal to or the *next-lower* contained timestamp compared to the request timestamp. We refer the interested reader to our previous work [8] for more details on this process. The B$^+$-Tree structure allows to perform this query efficiently with time complexity $\mathcal{O}(\log n)$ [**R1, R2**]. In addition to this primary index, ChronoDB also maintains a secondary index where the temporal keys are first ordered by timestamp and then by key in order to support time range queries [**R3, R4**].

3.2 Apache TinkerPop

The *TinkerPop* framework is the *de-facto* standard interface between applications and graph databases. It is designed in a modular fashion. The core module is the *property graph API* [23] which specifies the Java interfaces for vertices, edges, properties and other structural elements. In such a property graph, each vertex and edge can have properties which are expressed as key–value pairs. Another module is the graph query language *Gremlin*. In contrast to the property graph API, which is only a specification, Gremlin comes with a default implementation that is built upon the property graph API interfaces. Other modules include a standard *test suite* for TinkerPop vendors, and a generic server framework for graph databases called *Gremlin Server*.

4 Proposed Solution

In this section, we present our novel versioning concepts for TinkerPop Online Transaction Processing (OLTP) graphs. We give an overview over the architecture of ChronoGraph, outline how graph data is mapped to the underlying versioned key–value store, and how the versioning concepts affect this process.

4.1 ChronoGraph Architecture

Our open-source project ChronoGraph provides a TinkerPop-compliant graph database implementation with additional versioning capabilities. In order to achieve this goal, we employ a layered architecture as outlined in Fig. 2(a). In the remainder of this section, we provide an overview of this architecture in a bottom-up fashion.

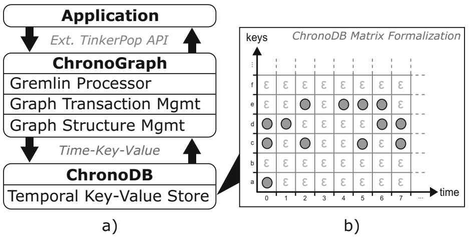


Fig. 2. ChronoGraph architecture [10].

The bottom layer of the architecture is a temporal key-value store, i.e. a system capable of working with *time–key–value* tuples as opposed to plain key–value pairs in regular key-value stores. For the implementation of ChronoGraph, we have chosen to use our own implementation called *ChronoDB*, which is also available as an open-source project on GitHub. Figure 2(b) shows the matrix formalization of the store, which we have presented in our previous work [8] and summarized in Sect. 3.

ChronoGraph itself consists of three major components. The first component is the *graph structure* management. It is responsible for managing the individual vertices and edges that form the graph, as well as their referential integrity. As the underlying storage mechanism is a key-value store, the graph structure management layer also performs the decomposition of the graph into key-value pairs and the conversion between the two formats. We present the technical details of this format in Sect. 4.2. The second component is the *transaction management*. The key concept here is that each graph transaction is associated with a timestamp on which it operates. Inside a transaction, any read request for graph content will be executed on the underlying storage with the transaction timestamp. ChronoGraph supports full ACID transactions with the highest possible isolation level ("serializable", also known as "snapshot isolation", as defined in the SQL Standard [13]). The underlying versioning system acts as an enabling technology for this highest level of transaction isolation, because any given version of the graph, once written to disk, is effectively immutable. All mutating operations are stored in the transaction until it is committed, which in turn produces a new version of the graph, with a new timestamp associated to it. Due to this mode of operation, we do not only achieve repeatable reads, but also provide effective protection from phantom reads, which is a common problem in concurrent graph computing. As the graph transaction management is heavily relying on the transactional capabilities of ChronoDB, we refer the interested reader to our previous work [8] for further details. The third and final component is the *query processor* itself which accepts and executes Gremlin queries on the graph system. As each graph transaction is bound to a timestamp, the query language (Gremlin) remains timestamp-agnostic, which allows the execution of any query on any desired timestamp [**R1**].

The application communicates with ChronoGraph by using the regular TinkerPop API, with additional extensions specific to versioning. The versioning itself is entirely transparent to the application to the extent where ChronoGraph can be used as a drop-in replacement for any other TinkerPop 3.x compliant implementation. The application is able to make use of the versioning capabilities via additional methods, but their usage is entirely optional and not required when working on tasks that do not involve history analysis. We discuss the details of the extended API in Sect. 4.4.

4.2 Data Layout

In order to store graph data in our Temporal Key–Value Store, we first need to disassemble the graph into parts that can be serialized as values and be addressed by keys. Then, we need to persist these pairs in the store. We will first discuss how we disassemble the graph, followed by an overview of the concrete key–value format and how versioning affects this process.

Partitioning: The Star Graph Format. Like many other popular graph databases, e.g. Titan DB, we rely on the *Star Graph* partitioning in order to disassemble the graph into manageable pieces.

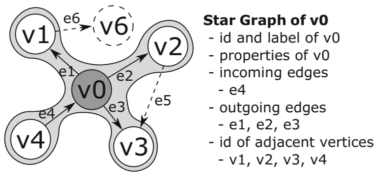


Fig. 3. Star graph partitioning by example.

Figure 3 shows an example of a star graph. A star graph is a subset of the elements of a full graph that is calculated given an origin vertex, in this case $v0$. The star graph contains all properties of the vertex, including the *id* and the *label*, as well as all incoming and outgoing edges (including their *label*, *id* and properties). All adjacent vertices of the origin vertex are represented in the star graph by their *id*s. Their attributes and remaining edges (indicated by dashed lines in Fig. 3) are not contained in the star graph of $v0$. This partitioning was chosen due to its ability to reconstruct the entire graph from disk without duplicating entire vertices or their attribute values. Furthermore it is suitable for lazy loading of individual vertices, as only the immediate neighbourhood of a vertex needs to be loaded to reconstruct it from disk.

Key–Value Layout. Starting from a star graph partitioning, we design our key–value layout. Since all graph elements in TinkerPop are mutable by definition and our persistent graph versions have to be immutable, we perform a bijective mapping step before persisting an element. We refer to the persistent, immutable version as a *Record*, and there is one type of record for each structural element in the TinkerPop API. For example, the mutable *Vertex* element is mapped to an immutable *VertexRecord*. A beneficial side-effect of this approach is that we hereby gain control over the persistent format, and can evolve and adapt each side of the mapping individually if needed. Table 2 shows the contents of the most important record types.

Table 2. TinkerPop API to record mapping [10].

TinkerPop	Record	Record contents
Vertex	VertexRecord	id, label,
		PropertyKey → PropertyRecord
		In: EdgeLabel → EdgeTargetRecord
		Out: EdgeLabel → EdgeTargetRecord
Edge	EdgeRecord	id, label,
		PropertyKey → PropertyRecord
		id of InVertex, id of OutVertex
Property	PropertyRecord	PropertyKey, PropertyValue
—	EdgeTargetRecord	id of edge, id of other-end Vertex

In Table 2, all *id* and *label* elements, as well as all *PropertyKey*s, are of type `String`. The *PropertyValue* in the *PropertyRecord* is assumed to be in byte array form. An arrow in the table indicates that the record contains a *mapping*, usually implemented with a regular hash map. An element that deserves special attention is the *EdgeTargetRecord* that does not exist in the TinkerPop API. Traversing from one vertex to another via an edge label is a very common task in a graph query. In a naive mapping, we would traverse from a vertex to an adjacent edge and load it, find the id of the vertex at the other end, and then resolve the target vertex. This involves two steps where we need to resolve an element by ID from disk which has a negative impact on query performance. However, we cannot store all edge information directly in a *VertexRecord*, because this would involve duplication of all edge properties on the other-end vertex. We overcome this issue by introducing an additional record type. The *EdgeTargetRecord* stores the id of the edge and the id of the vertex that resides at the "other end" of the edge. In this way, we can achieve basic vertex-to-vertex traversal in one step. At the same time, we minimize data duplication and can support edge queries (e.g. `g.traversal().E()` in TinkerPop), since we have the full *EdgeRecord*s as standalone elements. A disadvantage of this solution is the fact that we still need to do two resolution steps for any query that steps from vertex to vertex and has a condition on a *property* of the edge in between. This trade-off is common for graph databases, and we share it with many others, e.g. Neo4j. We will discuss this with a concrete example in Sect. 4.3.

For each record type, we create a *keyspace* in the underlying key–value store. We serialize the record elements into a binary sequence. This binary sequence serves as the value for the key–value pairs and the id of the element is used as the corresponding key. The type of record indicates which keyspace to use, completing the mapping to the key–value format. The inverse mapping involves the same steps: given an element ID and type, we resolve the key–value pair from the appropriate keyspace by performing a key lookup using the ID. Then, we deserialize the binary sequence, and apply our bijective element-to-record mapping in the inverse direction. When loading a vertex, the properties of the incoming and outgoing edges will be loaded *lazily*, because the EdgeTargetRecord does not contain this information and loading edge properties immediately would therefore require an additional lookup. The same logic applies to resolving the other-end vertices of EdgeTargetRecords, allowing for a lazy (and therefore efficient and RAM-conserving) solution.

4.3 Versioning Concept

When discussing the mapping from the TinkerPop structure to the underlying key–value store in Sect. 4.2, we did not touch the topic of versioning. This is due to the fact that our key–value store *ChronoDB* is performing the versioning on its own. The graph structure does not need to be aware of this process. We still achieve a fully versioned graph, an immutable history and a very high degree of sharing of common (unchanged) data between revisions. This is accomplished by attaching a fixed *timestamp* to every *graph transaction*. This timestamp is always the same as in the underlying ChronoDB transaction. When reading graph data, at some point in the resolution process we perform a *get(...)* call in the underlying key–value store, resolving an element (e.g. a vertex) by ID. At this point, ChronoDB uses the timestamp attached to the transaction to perform the temporal resolution. This will return the value of the given key, at the specified timestamp. For details on the temporal resolution process, we refer the interested reader to our previous work dedicated to ChronoDB [8].

In order to illustrate this process, we consider the example in Fig. 4. We open a transaction at timestamp 1234 and execute the following Gremlin query:

```
V("v0").out("e0").outE("e1").has("p", "x").inV()
```

We start by resolving the vertex *v0* from the database. Since our transaction uses timestamp 1234, ChronoDB will look up the temporal key *v0@1234*, and

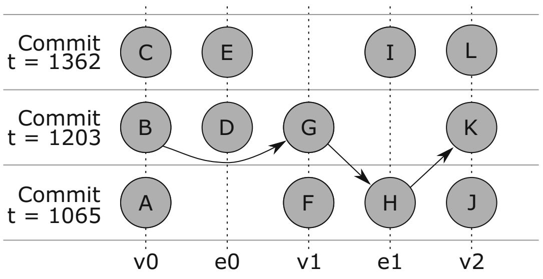


Fig. 4. Example: Navigating in a graph version.

return the value labelled as B in Fig. 4[7]. Value A is not visible because it was overwritten by B at timestamp 1203, and value C is also not visible because it was written *after* our transaction timestamp. Next, we navigate the outgoing edge labelled as *e0*. Our store does contain information on that edge, but since the query does not depend on any of its properties, we use the *EdgeTargetRecord* stored in B and directly navigate to *v1*. We therefore ask ChronoDB for the value associated with temporal key *v1@1234*, and receive value G. For the next query step, we have a condition on the outgoing edge *e1*. Our EdgeTargetRecord in value G does not contain enough information to evaluate the condition, hence we need resolve the edge from the store. Querying the temporal key *e1@1234* will return the value H, which is shared with the previous version because it was not changed since then. After evaluating the condition that the property “p” on edge version H is indeed set to the value “x” (as specified in the query), we continue our navigation by resolving the target of *e1*, which is *v2*. The temporal key *v2@1234* will result in the value K being returned.

Note that this final navigation step starts at an element that was reused from the commit at timestamp 1065 and ends at the state of *v2* that was produced by the commit at timestamp 1203. This is possible because graph elements refer to each other by ID, but these references do not include the timestamp. This information is injected from the transaction at hand, allowing for this kind of navigation and data reuse. This is a major step towards fulfilling requirement [**R1**]. As ChronoDB offers logarithmic access time to any key-value pair on any version, this is also in line with requirement [**R2**].

4.4 TinkerPop Compatibility

The Apache TinkerPop API is the *de-facto* standard interface between graph databases and applications built on top of them. We therefore want Chrono-Graph to implement and be fully compliant to this interface as well. However, in order to provide our additional functionality, we need to extend the default API at several points. There are two parts to this challenge. The first part is *compliance* with the existing TinkerPop API, the second part is the *extension* of

[7] All circles in Fig. 4 represent serialized vertex records or edge records.

this API in order to grant access to new functionality. In the following sections, we will discuss these points in greater detail.

TinkerPop API Compliance. As we described in Sects. 4.2 and 4.3, our versioning approach is entirely transparent to the user. This eases the achievement of compliance to the default TinkerPop API. The key aspect that we need to ensure is that every transaction receives a proper timestamp when the regular `g.tx().open()` method is invoked (see Listing 1.1). In a non-versioned database, there is no decision to make at this point, because there is only one graph in a single state. The logical choice for a versioned graph database is to return a transaction on the current *head* revision, i.e. the timestamp of the transaction is set to the timestamp of the latest commit. This aligns well with the default TinkerPop transaction semantics—a new transaction t should see all changes performed by other transactions that were committed before t was opened. When a commit occurs, the changes are always applied to the head revision, regardless of the timestamp at hand, because history states are immutable in our implementation in order to preserve traceability of changes. As the remainder of our TinkerPop implementation, in particular the query language Gremlin, is unaware of the versioning process, there is no need for further specialized efforts to align versioning with the TinkerPop API.

We employ the TinkerPop *Structure Standard Suite*, consisting of more than 700 automated JUnit tests, in order to assert compliance with the TinkerPop API itself. This test suite is set up to scan the declared *Graph Features* (i.e. optional parts of the API), and enable or disable individual tests based on these features. With the exception of *Multi-Properties*[8] and the *Graph Computer*[9], we currently support all optional TinkerPop API features, which results in 533 tests to be executed. We had to manually disable 8 of those remaining test cases due to problems within the test suite, primarily due to I/O errors related to file names generated by the test suite which are syntactically invalid on our Windows-based development system. The remaining 525 tests all pass on our API implementation.

TinkerPop Extensions. Having asserted conformance to the TinkerPop API, we created custom extensions that give access to the features unique to ChronoGraph. As the query language Gremlin itself remains completely untouched in our case, and the graph structure (e.g. `Vertex` and `Edge` classes) is unaware of the versioning process (as indicated in Sect. 4.3), we are left with one possible extension point, which is the `Graph` interface itself. In order to fulfill requirements [**R1**] and [**R2**], we need to add a method to open a transaction on a user-provided timestamp. By default, a transaction in TinkerPop on a `Graph` instance `g` is opened via one of the following methods:

[8] We do not support multi-valued properties directly as intended by TinkerPop. However, we do support regular properties of List or Set types.

[9] The Graph Computer is the entry point to the Online Analytics Processing (OLAP) API. Support for this feature may be added in future versions of ChronoGraph.

```
// Variant A: Thread-bound transaction
g.tx().open();
// Variant B: Unbound transaction
Graph txGraph = g.tx().createThreadedTx();
```

Listing 1.1. Opening TinkerPop Transactions.

We expanded the `Transaction` class by adding two new overrides to the `open(...)` and `createThreadedTx(...)` methods:

```
Date userDate = ... ;   long userTime = ...;
// Variant A:
graph.tx().open(userDate);
graph.tx().open(userTime);
// Variant B:
Graph tx1 = g.tx().createThreadedTx(userDate);
Graph tx2 = g.tx().createThreadedTx(userTime);
```

Listing 1.2. Opening ChronoGraph Transactions.

Using these additional overrides, the user can decide the `java.util.Date` or `java.lang.Long` timestamp on which the transaction should be based. This small change of adding an additional time argument is all it takes for the user to make full use of the time travel feature, the entire remainder of the TinkerPop API, including the structure elements and the Gremlin query language, behave as defined in the standard. With these additional methods, in combination with the technical details presented in the previous sections, we fully cover requirements [**R1**] and [**R2**].

In order to provide access to the history of a single Vertex or Edge, we added the following methods to our `Graph` implementation:

```
Vertex v = ... ;     Edge e = ... ;
Iterator<Long> it1 = graph.getVertexHistory(v);
Iterator<Long> it2 = graph.getEdgeHistory(e);
```

Listing 1.3. Accessing Graph Element History.

These methods allow access to the history of any given edge or vertex. The history is expressed by an `Iterator` over the change timestamps of the element in question, i.e. whenever a commit changed the element, its timestamp will appear in the values returned by the iterator. Each of these values can then be used as an argument for calling `g.tx().open(...)` in order to retrieve the state of the element at the desired point in time. The implementation of the history methods delegate the call directly to the underlying ChronoDB, which retrieves the history of the key–value pair associated with the ID of the given graph element. This history is extracted from the primary index, which is first sorted by key (which is known in both scenarios) and then by timestamp. This ordering allows the two history operations to be very efficient as resolving the element ID only requires a lookup in logarithmic time, followed by backwards iteration over the primary index (i.e. iteration over change timestamps) until a different element ID is encountered (c.f. Table 1).

The final requirement with respect to versioning capabilities is the demand for an operation that lists all changes within a given time range, regardless of the affected elements [**R4**]. In order to meet this requirement, we added the following methods to our `Graph` implementation:

```
long from = ...; long to = ...;
Iterator<TemporalKey> it1, it2;
it1 = graph.getVertexModificationsBetween(from, to);
it2 = graph.getEdgeModificationsBetween(from, to);
```

Listing 1.4. Listing Changes within a Time Range.

These methods grant access to iterators that return *TemporalKeys*. These keys are pairs of actual element identifiers and change timestamps. Just as their element-specific counterparts, it is intended that these timestamps are used for opening transactions on them in order to inspect the graph state. Combined calls to `next()` on `it1` and `it2` will yield the complete list of changes upon iterator exhaustion, fulfilling requirement [**R4**]. Analogous to their element-specific counterparts, these methods redirect directly to the underlying ChronoDB instance, where a secondary temporal index is maintained that is first ordered by timestamp and then by key. This secondary index is constructed per keyspace. Since vertices and edges reside in disjoint keyspaces, these two operations do not require further filtering and can make direct use of the secondary temporal index.

Transaction Semantics. The Apache TinkerPop API is currently available in its third (major) version. It evolved alongside its implementations, which range from local graphs (e.g. the in-memory reference implementation TinkerGraph) to highly distributed systems (e.g. Titan DB). Due to this diversity, the requirements towards transaction semantics, in particular behaviour under concurrent access, are specified very loosely in TinkerPop itself. For example, when iterating over the outgoing edges of a vertex, TinkerPop only specifies that the iteration itself should never return a `null` value and should never throw a `ConcurrentModificationException`, but details regarding the visibility of changes made by other, concurrent transactions are unspecified.

Since the reference implementation TinkerGraph, which is provided alongside the API, does not support transactions[10], we had to design the transaction semantics by ourselves. When we implemented ChronoDB, we envisioned it to be a system suitable for storing data for analysis purposes, therefore the consistency of a view and the contained data is paramount. As all stored versions are effectively immutable, we chose to implement a full ACID transaction model in ChronoDB with the highest possible isolation level ("Serializable" [13]). As ChronoGraph is based on ChronoDB, it follows the same transaction model. To the best of our knowledge, ChronoGraph is currently the only implementation of the TinkerPop API v3.x that is full ACID in the strict sense, as many others opt for *repeatable reads* isolation (e.g. OrientDB) while ChronoGraph supports *snapshot* isolation. A proposal for snapshot isolation for Neo4j was published recently [20], but it is not part of the official version. Graph databases without ACID transactions and snapshot isolation often suffer from issues like *Ghost*

[10] Transaction support is an optional feature in TinkerPop 3.

Vertices[11] or *Half Edges*[12] which can cause inconsistent query results and are very difficult to handle as an application developer. These artifacts are negative side-effects of improper transaction isolation, and application developers have to employ techniques such as soft deletes (i.e. the addition of "deleted" flags instead of true element deletions) in order to avoid them. This introduces additional complexity in the application and also reduces query performance. As ChronoGraph strictly adheres to the ACID principles, these inconsistencies can not appear by design.

5 Related Work

The idea to have a versioned graph database is well-known. Several authors proposed different approaches [3, 26, 27] which also highlights the importance of the problem. However, our project is different from existing solutions in one key aspect: In contrast to other authors, we do not propose to implement the versioning capabilities within a graph itself, but rather on a lower level. This section is dedicated to a discussion of the resulting advantages and disadvantages of this design choice.

5.1 Functionality and Usage Implications

Implementing versioning of a graph within an existing graph database is a tempting idea, given that a considerable implementation and quality assurance effort has been put into modern graph databases like Neo4j. Early works in this area date back to 2013 when Castelltort and Laurent published their work on this topic [3]. Further work in the same vein was published recently by Taentzer et al. [29]. However, as Castelltort's and Laurent's paper clearly shows, the versioned "meta-graph" structure can become very complex even for small sets of applied changes. Changes in vertex properties are easy to represent by maintaining several copies of the vertex and linking them together by "predecessor" edges, but dealing with structural changes, in particular the addition or removal of edges, is difficult to represent within a graph structure itself. Due to this added complexity in structure, the resulting queries need to become more sophisticated as well. This can be mitigated by implementing a query translation mechanism that takes a regular graph query and a timestamp as input and transforms it into a query on the meta-graph. Aside from the inherent complexity of this mapping, the performance of the resulting transformed query will suffer inevitably, as the overall graph structure is much larger than a single graph version would be. In the common case where a query should be executed on one particular timestamp, the amount of irrelevant data (i.e. the number of graph elements that do

[11] Vertices that have been deleted by transaction *t1* while being modified concurrently by transaction *t2* do not disappear from the graph; they remain as *Ghost*s.

[12] Half Edges refer to the situation where an edge is only traversable and visible in one direction, i.e. the *out*-vertex lists the edge as outgoing, but the *in*-vertex does not list it as incoming, or vice versa.

not belong to the requested revision) in the graph increases linearly with every commit. A non-versioned general purpose graph database such as Neo4j, being unaware of the semantics of versioning, has no access to any means for reacting to and/or devising a countermeasure against this problem.

An additional issue arises when many edges in the meta-graph, which are introduced by the versioning process, point to the same vertex. Vertices with a high number of incoming and/or outgoing edges are commonly referred to as *super vertices*, and represent a problematic corner case for many graph database implementations in terms of performance and storage. Given a suitable set of graph changes, most "version-to-graph" mappings are introducing such super vertices in order to realize graph versioning.

Other related approaches e.g. by Semertzidis and Pitoura [26,27] or by Han et al. [11], assume the existence of a series of graph snapshots as input to their solutions. These approaches do not aim for online transaction processing (OLTP) capabilities and focus on the analysis of a series of static graphs. A direct comparison with our approach is therefore not feasible. However, the data managed by ChronoGraph may serve as an input to those tools, as each graph revision can be extracted individually and consequently be treated as a series of snapshots.

Our implementation is a stark contrast to existing solutions. We implement the versioning process at a lower level, in a generic temporal key–value store called ChronoDB. This store is aware of the semantics of the versioning process, and is capable of solving the problem of long histories [9], unlike the previously mentioned solutions. There is no particular mapping required in order to achieve graph versioning, in fact our mapping is very similar to the one employed by Titan DB. Therefore, no additional auxiliary graph elements (and in particular no super vertices) are introduced for the purpose of versioning. To the end user, the versioning process is completely transparent, as our implementation is fully compliant with the standard TinkerPop API for non-versioned graphs. There is no need for translating one graph query into another in order to run it on a different graph version. A developer familiar with the TinkerPop API can start using ChronoGraph without any particular knowledge about the versioned nature of the graph. By offering additional methods, which are very much in line with the concepts and intentions of the TinkerPop API, we grant access to the temporal features. Additionally, ChronoGraph is fully ACID compliant with snapshot isolation for concurrent transactions, preventing common artifacts that arise in other, non-ACID graph databases, such as ghost vertices and half edges. Our solution is strongly based on immutability of existing versions, which aids in preserving traceability of changes and allows extensive disk-level sharing of data that remained unchanged between revisions.

5.2 Limitations and Drawbacks

Our approach is tailored towards the use case of having a *versioned* graph (as opposed to a *temporal* graph), which entails that queries on a *single* timestamp are the prevalent form of read access. Even though we support additional auxiliary methods for traversing the history of a single vertex or edge, and listing all changes within a given time range, our approach is far less suitable for use cases with an emphasis on temporal analysis that require time range queries, or detection of patterns on the time axis (as in graph stream analysis [18,21]). For example, answering the question “Which elements often change together?”, while possible in our solution, can not be implemented in an efficient way that does not require linear scanning through the commit logs. Another example would be the query “List all vertices that have ever been adjacent to a given one”, which would again involve linear iteration in our solution. In general, our graph is a TinkerPop implementation and therefore optimized with the traversal language *Gremlin* in mind. As such, it does not lend itself as well to declarative, pattern-driven search approaches like *Cypher* as a dedicated Cypher graph implementation (e.g. Neo4j) would do.

We are currently also not offering any means for distributing the graph among multiple machines (see Sect. 7 for details). This limits the scale of our graph to sizes manageable within the physical memory and computing resource restrictions of a single machine. Currently, the largest ChronoGraph instance used in practice that we are aware of has about 500.000 elements (vertices plus edges) in the head revision. This number does not include past revisions of elements. An essential drawback of our solution is that, due to the versioned nature of our data, we cannot rely as much on dictionaries with $\mathcal{O}(1)$ access times (e.g. Hash Maps) as regular general purpose graph databases, because of the temporal resolution steps that happen during every navigation step. Those steps have a complexity of $\mathcal{O}(\log(n))$, which also limits the scalability of our graph.

Finally, we have to acknowledge the fact that ChronoGraph is an ongoing work-in-progress research project, therefore the limit regarding optimizations in the code is by far not reached yet.

6 Evaluation

ChronoGraph is a unique addition to the TinkerPop family. Table 3 shows a very high-level look at the most relevant TinkerPop implementations in practice and compares them to ChronoGraph. Titan, Neo4j and ChronoGraph are supporting the new 3.x versions of TinkerPop, while OrientDB is still using the 2.x version. The remainder of the table can be divided into two parts. On the one hand there are distributed systems that are intended for deployment on multiple machines, and on the other hand there are systems deployed locally on one machine. In the distributed domain, the BASE[13] approach is prevalent due to its weaker consistency guarantees. For local deployments, all graph databases in the list follow the ACID principles. However, they all implement them in different ways. In particular the *Isolation* property allows for a certain degree of freedom in interpretation. Most local graph databases listed in Table 3 have *Read Committed* isolation, which is the second weakest of the four levels identified in the SQL 2011 Standard [13]. OrientDB in addition provides *Repeatable Reads* isolation exclusively in local mode, but ChronoGraph is the only listed implementation

[13] **B**asically **A**vailable, **S**oft State, **E**ventual Consistency.

Table 3. Comparison of TinkerPop implementations [10].

Graph-Database		TinkerPop version	Deployment	Paradigm	Max. isolation level
ChronoGraph		3.x	Local	ACID	Snapshot
Titan	on BerkeleyDB	3.x	Local	ACID	Read committed
	on Cassandra	3.x	Distributed	BASE	n/a
	on HBase	3.x	Distributed	BASE	n/a
Neo4j		3.x	Distributed	BASE/ACID	Read committed
OrientDB	Local mode	2.x	Local	ACID	Repeatable reads
	Distributed mode	2.x	Distributed	ACID	Read committed

that provides true *Snapshot* isolation, which is the highest possible isolation level.

Our open-source implementation of ChronoGraph serves as our proof-of-concept and will be used for the evaluation of the presented concepts. With its novel transparent versioning capabilities and full ACID snapshot isolation transactions, it offers a unique set of features and improves upon the state-of-the-art in TinkerPop OLTP graphs. We assert the conformance to the TinkerPop standard via the automated test suite provided by TinkerPop which encompasses around 700 test cases. We furthermore assert the correctness of our additional functionality by adding another 1400 automated test cases (which are also published alongside the source code), achieving a total statement coverage of approximately 70%. In order to demonstrate that the impact on performance of these new features does not harm the overall applicability of our graph database, we conducted a comparative experiment with the top three major TinkerPop implementations on the market, alongside a second experiment that demonstrates the versioning capabilities of ChronoGraph.

6.1 Experiment Setups

This section presents two experiments with their respective setups and results. The comparative performance experiment evaluates ChronoGraph against other popular graph databases, while the version history growth experiment focuses on the versioning capabilities of ChronoGraph and demonstrates the impact on performance as more revisions are added to a ChronoGraph instance.

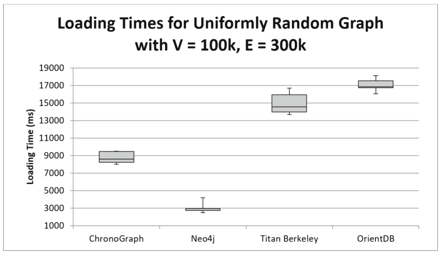
All benchmarks were executed with 1.5 GB (comparative experiment) or 3.0 GB (history growth experiment) of RAM available to the JVM on an Intel Core i7-5820K processor with 3.30 GHz and a Crucial CT500MX200 SSD.

Comparative Performance Experiment. We generated a uniformly random graph with 100.000 vertices and 300.000 edges to connect them. We then selected a random subset of 10.000 vertices. In order to assert the reproducibility of the experiment, we persisted both the adjacency list and the identifiers of the vertex subset in a plain text file. Using the graph database under test (GUT), we then loaded the text file, created the corresponding TinkerPop structure and persisted it as a new graph in the GUT. In order to have a common baseline, we avoided the use of vendor-specific batch-loading capabilities. The time it takes to perform this task is shown in Fig. 5(a). The displayed box plot shows the results over 10 runs.

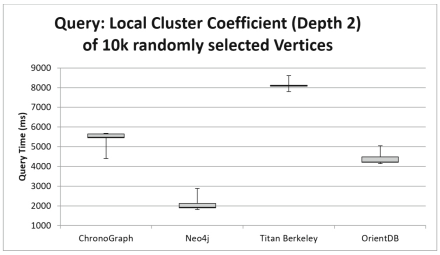
After importing the graph, we loaded the list of 10.000 random vertex IDs from the text file. For each vertex ID, we fetched the corresponding vertex from the GUT and calculated its *local cluster coefficient* as follows:

1. Calculate the set of all edges which are reachable from the source vertex within two steps, ignoring edge directions. We call this set the *neighbourhood*.
2. If the neighbourhood has size 2 or less, we immediately return zero. Otherwise, we continue with the next step.
3. For each vertex in the neighbourhood, we calculate the number of existing edges which start and end at a vertex within the neighbourhood, and count them.
4. We divide the resulting number by the number of edges the neighbourhood would have if it were a fully connected clique, and return this result.

This experiment design is very similar to the experiment presented by Ciglan et al. [4] and covers two of the primary capabilities of graph databases, which are random access and neighbourhood navigation. Figure 5(b) shows the results of this benchmark, accumulated over 10 runs.



(a) Graph Loading Times



(b) Query Response Times

Fig. 5. Performance of TinkerPop graph creation and querying [10].

As Fig. 5 clearly indicates, Neo4j is the fastest competitor in both loading and querying the graph by a wide margin. We would like to emphasize that we included Neo4j in this benchmark for reference purposes due to its popularity; its feature set is hardly comparable to the other databases in the benchmark as it primarily follows the BASE principle which allows for a wide range of optimizations that would not be possible in an ACID environment. Our ChronoGraph implementation takes second place in loading speed. The reason for that is likely due to the fact that OrientDB first needs to convert the graph elements into its internal document representation, while Titan is limited by the insertion speed of the underlying Berkeley DB. In terms of read access speed, ChronoGraph is the middle ground between OrientDB and Titan Berkeley in this benchmark.

Here, we only deal with a single graph version, therefore a comparable speed to non-versioned graph databases is to be expected. We still suffer from a slight overhead in calculation due to the present versioning engine, which is the reason why OrientDB performs better. The main use case of Titan is in distributed environments, which explains why restricting it to a single Berkeley DB instance (the only officially supported ACID backend) causes a degradation in performance.

Version History Growth Experiment. In a second experiment we analyze the performance impact as an increasing amount of versions are added to the system via regular commits. The goal of this experiment is to demonstrate that the impact of long histories does not impede the practical applicability of our database. For this experiment, we assume frequent small-scale changes. Even though ChronoGraph supports versioning for vertex and/or edge property values, in this experiment we only consider changes that alter the graph topology, because they are more complex to handle for the versioning engine than basic property changes (as outlined in Sect. 5).

We employ the same local cluster coefficient calculation query as in the previous experiment on a graph with an increasing number of versions. To eliminate the impact of an increase in graph size, we choose the modifications randomly from a probability distribution which is calculated on the fly to increase the graph size if too many deletions have occurred, and to reduce it again if too many elements have been added. All operations preserve the uniform randomness of the graph structure. The four distinct events are vertex addition, vertex removal, edge addition and edge removal. The removal operations pick their target element at random. When adding a vertex, we also connect it to another, randomly chosen existing vertex via a new edge. A commit can contain any combination of these operations.

In this experiment, each commit consists of 50 such structural changes. As in the previous experiment, the initial graph is uniformly random and has 100.000 vertices and 300.000 edges. The modifications in the commits alter these values. We use the random distribution of changes as a tool to keep the graph from becoming too large or too small, as we are only interested in the impact of a growing history and want to eliminate the influence of a growing or shrinking graph structure. Both the number of vertices and the number of edges is bounded within $\pm 5\%$ of the original numbers. The probability distribution for the change events is recalculated after every modification.

The experiment algorithm operates in iterations. In each iteration, it performs 300 commits, each of them containing 50 operations as outlined above. Then, a timestamp is picked on which a graph transaction is opened, and the local cluster coefficient query from the first experiment is calculated for 10.000 randomly selected vertices. We measure the accumulated time of these queries and repeat each measurement 10 times (keeping the timestamp constant, but selecting a different, random set of vertices each time). We would like to emphasize that the time for opening a transaction is independent of the chosen timestamp, i.e. opening a transaction on the head revision is equally fast as opening a transaction on the initial revision. The timestamp selection is done in four different ways, which give rise to the four different graphs in Fig. 6:

– INITIAL: Always uses the timestamp of the initial commit.
– HEAD: Always uses the timestamp of the latest commit.
– MIDWAY: Uses the timestamp at the arithmetic mean of the initial and latest commit timestamps.
– RANDOM: Randomly chooses a timestamp between the initial and latest commit (inclusive).

We granted 3 GB of RAM to the JVM in this experiment (even though the benchmark program can be successfully executed with less memory) in order to prevent excessive garbage collection overhead. Compared to the first experiment, we are faced with much larger volumes of data in this case, and the test code itself has to manage a considerable amount of metadata, e.g. in order to assert the correct calculation of the probability distributions.

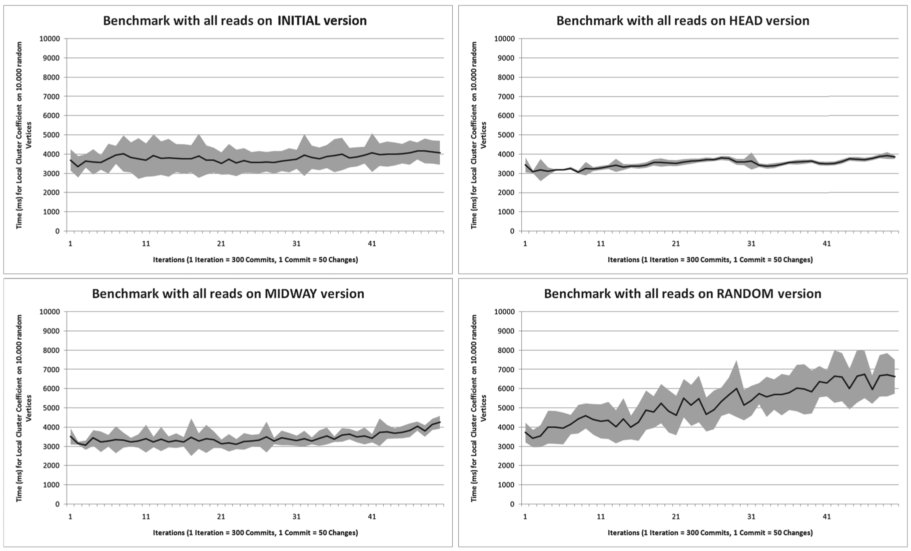


Fig. 6. Query performance with increasing number of versions [10].

Figure 6 showcases a number of interesting properties of ChronoGraph. First of all, given a graph that retains an almost constant size over time, the number of versions has a very minor effect on the query performance if the INITIAL or HEAD revisions are requested. The present increase is due to the larger number of elements in the underlying B$^+$-Tree structure in ChronoDB. The INITIAL version has a notably higher standard deviation than the HEAD version. This is due to the experiment setup. As new changes are committed, they are written through our versioning-aware cache. The HEAD case can take full advantage of this fact. However, for the INITIAL case, the write through eliminates older entries from the Least-Recently-Used cache which have to be fetched again to answer the queries on the initial commit timestamp while the newly-written entries from the head revision are never queried.

The MIDWAY version suffers from a similar problem as the INITIAL version, however to a lesser extent because the request timestamp is closer to the latest commit than in the INITIAL case. We would like to emphasize that, while the underlying store offers the same performance for any version, the temporal cache is limited in size and cannot hold all entries in the experiment. Therefore, choosing a timestamp that is closer to the latest commit causes an improvement in performance due to the write-through mechanics of our commit operations.

Finally, the worst case scenario is the RANDOM variant of our experiment. Here, each read batch chooses a different timestamp at random from the range of valid timestamps. This causes a considerable number of cache misses, increasing the standard deviation and query response time. As outlined in Sect. 3, the underlying data structure in our store is a B^+-Tree that contains all revisions, which is the reason for the resulting logarithmic curve in Fig. 6.

We would like to emphasize that the versioning engine is faced with considerable volumes of changes in this experiment. After 50 iterations, the version history consists of 15,000 individual graph revisions with 50 modifications each. This translates into 750.000 high-level changes in addition to the initial version, which already contained 400.000 graph elements. Considering that a graph structure change translates into several key-value pair changes (e.g. a vertex deletion cascades into the deletion of all connected edges, which cascades into adjacency list changes in the vertices at the other end), the number of atomic changes is even higher. Each of these atomic changes must be tracked in order to allow for per-element history queries. If we go by the realistic assumption that each high-level graph change on average entails 3 changes in the underlying key–value store, this results in a store that has more than five times the number of elements compared to the initial graph. As Fig. 6 shows, this increase in data volume does not entail an equivalent increase query response times, which demonstrates the scalability of our approach with a high number of revisions [**R2**].

Evaluation Result Summary. The additional level of isolation, the strict adherence to the ACID properties and the versioning capabilities are the features that set apart ChronoGraph from all other existing TinkerPop graph databases. Even though these features require an undeniable overhead in calculation times, we have shown in the performance experiments that this overhead has no severe impact in the overall applicability of our database. We consider this overhead to be a small price to pay considering the gained benefits of versioning and ACID transactions. We have also shown that the performance of our ChronoGraph implementation can compete with (and in some cases even outperform) existing non-versioned solutions.

7 Outlook and Future Work

The existing versioning capabilities are based on the underlying assumption that the history of each element is immutable, i.e. the past does not change. This principle will considerably ease the distribution of the versioned graph data among multiple machines, as newly incoming commits can only alter the database contents in the head revision in an append-only fashion. This allows for highly effective caching and data replication without risking to ever encounter stale data. We plan on implementing such capabilities in ChronoGraph in the future.

Another feature which we intend to support in upcoming releases is the TinkerPop `GraphComputer` interface. It is a generalized interface for computation on distributed graphs, geared towards online analytics processing (OLAP) by utilizing map–reduce patterns and message passing. Thanks to the abstraction layer provided by Gremlin, this will also allow our users to run queries in a variety of languages on our graph, including SPARQL[14] [1]. In general, Graph Computer implementations are responsible for asserting repeatable reads (as their underlying graph might not provide this guarantee), but the fact that we are working on a full ACID graph should ease the implementation considerably.

Finally, we are also applying ChronoGraph in an industrial context in the IT Landscape Documentation tool Txture[15]. In this collaboration, we have successfully connected high-level modelling techniques with our versioned graph database. Txture assists data centers in the management of their IT assets, and in particular their interdependencies. At its core, Txture is a repository for these asset models, and ChronoGraph acts as the database of this repository, providing the required query evaluation, persistence and versioning capabilities. Txture is currently in use by several industrial customers, including the data center Allgemeines Rechenzentrum (ARZ) and a globally operating semiconductor manufacturer. We will publish a case study on the role of ChronoGraph within Txture in the near future.

8 Summary

This paper presented ChronoGraph, our TinkerPop-compliant graph database with transparent system-time versioning support. We provided an analysis of the requirements we considered for our approach, such as the ability to execute any query on any graph version, or to trace the history of individual graph elements. We then introduced the technical background of our work in Sect. 3, focusing on the employed software and terminology. Section 4 outlined the architecture of our solution and highlighted the key aspects which allow ChronoGraph to meet the aforementioned requirements. We explained in detail how the graph structure is mapped to the underlying versioned key-value store, and how navigational queries traverse the resulting data structures. We also showcased how our implementation aligns with the TinkerPop standard and the extensions that

[14] https://www.w3.org/TR/sparql11-query/.
[15] www.txture.io.

are required in order to make our additional non-standard features accessible for the application programmer. In Sect. 5 we compared our concepts to existing approaches and discussed advantages, drawbacks and distinguishing features of individual solutions. Section 6 followed up on the discussion with controlled experiments. The results clearly show that ChronoGraph offers competitive performance in comparison to other popular graph databases, and in particular provide a strong indication that the overhead introduced by the versioning engine does not harm the overall applicability of our implementation in practice. Finally, we discussed our roadmap for future research and development in Sect. 7.

We made two key contributions in this paper. The first contribution is the addition of our novel versioning concepts to the state of the art of graph databases with the additional benefit of achieving snapshot-level transaction isolation and conformance to all ACID properties. The second contribution is the publication of the ChronoGraph source code which is freely available under an open-source license. This implementation has also been recognized by the Apache TinkerPop community as an official implementation of the standard.

References

1. Barbieri, D.F., Braga, D., Ceri, S., Valle, E.D., Grossniklaus, M.: C-SPARQL: a continuous query language for RDF data streams. Int. J. Seman. Comput. **4**(1), 3–25 (2010)
2. Becker, B., Gschwind, S., Ohler, T., Seeger, B., Widmayer, P.: An asymptotically optimal multiversion B-Tree. VLDB J. **5**(4), 264–275 (1996)
3. Castelltort, A., Laurent, A.: Representing history in graph-oriented NoSQL databases: a versioning system. In: Eighth International Conference on Digital Information Management (ICDIM 2013), Islamabad, Pakistan, 10–12 September 2013, pp. 228–234 (2013)
4. Ciglan, M., Averbuch, A., Hluchy, L.: Benchmarking traversal operations over graph databases. In: 2012 IEEE 28th International Conference on Data Engineering Workshops (ICDEW), pp. 186–189. IEEE (2012)
5. Easton, M.C.: Key-sequence data sets on indelible storage. IBM J. Res. Dev. **30**(3), 230–241 (1986)
6. Farwick, M., Trojer, T., Breu, M., Ginther, S., Kleinlercher, J., Doblander, A.: A case study on textual enterprise architecture modeling. In: 2013 17th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW), Vancouver, BC, pp. 305–309. IEEE (2013)
7. Felber, P., Pasin, M., et al.: On the support of versioning in distributed key-value stores. In: IEEE SRDS 2014, Nara, Japan, 6–9 October 2014, pp. 95–104 (2014)
8. Haeusler, M.: Scalable versioning for key-value stores. In: DATA 2016 - Proceedings of 5th International Conference on Data Management Technologies and Applications, Lisbon, Portugal, 24–26 July 2016, pp. 79–86 (2016)
9. Haeusler, M., Breu, R.: Sustainable management of versioned data. In: Proceedings of the 24th PhD Mini-Symposium. Budapest University of Technology and Economics (2017)
10. Haeusler, M., Nowakowski, E., Farwick, M., Breu, R., Kessler, J., Trojer, T.: ChronoGraph - versioning support for OLTP TinkerPop graphs. In: Proceedings of the 6th International Conference on Data Science, Technology and Applications - Volume 1: DATA, pp. 87–97. INSTICC, SciTePress (2017)

11. Han, W., Miao, Y., Li, K., Wu, M., Yang, F., Zhou, L., Prabhakaran, V., Chen, W., Chen, E.: Chronos: a graph engine for temporal graph analysis. In: Proceedings of the Ninth European Conference on Computer Systems, p. 1. ACM (2014)
12. Hart, M., Jesse, S.: Oracle Database 10G High Availability with RAC, Flashback & Data Guard, 1st edn. McGraw-Hill Inc., New York (2004)
13. ISO. SQL Standard 2011 (ISO/IEC 9075:2011) (2011)
14. Jensen, C.S., et al.: The consensus glossary of temporal database concepts — February 1998 version. In: Etzion, O., Jajodia, S., Sripada, S. (eds.) Temporal Databases: Research and Practice. LNCS, vol. 1399, pp. 367–405. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0053710
15. Lomet, D., Barga, R., Mokbel, M., Shegalov, G.: Transaction time support inside a database engine. In: Proceedings of the 22nd ICDE, p. 35, April 2006
16. Lomet, D., Hong, M., Nehme, R., Zhang, R.: Transaction time indexing with version compression. Proc. VLDB Endow. **1**(1), 870–881 (2008)
17. Lomet, D., Salzberg, B.: Access methods for multiversion data. SIGMOD Rec. **18**(2), 315–324 (1989)
18. McGregor, A.: Graph stream algorithms: a survey. SIGMOD Rec. **43**(1), 9–20 (2014)
19. Nascimento, M.A., Dunham, M.H., Elmasri, R.: M-IVTT: an index for bitemporal databases. In: Wagner, R.R., Thoma, H. (eds.) DEXA 1996. LNCS, vol. 1134, pp. 779–790. Springer, Heidelberg (1996). https://doi.org/10.1007/BFb0034730
20. Patiño Martínez, M., Sancho, D., Jiménez Peris, R., Brondino, I., Vianello, V., Dhamane, R.: Snapshot isolation for Neo4j. In: Advances in Database Technology (EDBT). OpenProceedings.org (2016)
21. Pigné, Y., Dutot, A., Guinand, F., Olivier, D.: GraphStream: a tool for bridging the gap between complex systems and dynamic graphs. In: Emergent Properties in Natural and Artificial Complex Systems. Satellite Conference within the 4th European Conference on Complex Systems (ECCS 2007), volume abs/0803.2 (2008)
22. Ramaswamy, S.: Efficient indexing for constraint and temporal databases. In: Afrati, F., Kolaitis, P. (eds.) ICDT 1997. LNCS, vol. 1186, pp. 419–431. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-62222-5_61
23. Rodriguez, M.A., Neubauer, P.: The graph traversal pattern. In: Graph Data Management: Techniques and Applications, pp. 29–46 (2011)
24. Salzberg, B.: File Structures: An Analytic Approach. Prentice-Hall Inc., Upper Saddle River (1988)
25. Saracco, C., Nicola, M., Gandhi, L.: A matter of time: temporal data management in DB2 10. IBM developerWorks (2012)
26. Semertzidis, K., Pitoura, E.: Durable graph pattern queries on historical graphs. In: Proceedings of the IEEE ICDE (2016)
27. Semertzidis, K., Pitoura, E.: Time traveling in graphs using a graph database. In: Proceedings of the Workshops of the (EDBT/ICDT) (2016)
28. Snodgrass, R.T.: Temporal databases. Computer **19**, 35–42 (1986)
29. Taentzer, G., Ermel, C., Langer, P., Wimmer, M.: A fundamental approach to model versioning based on graph modifications: from theory to implementation. Softw. Syst. Model. **13**(1), 239–272 (2014)
30. Tanase, I., Xia, Y., et al.: A highly efficient runtime and graph library for large scale graph analytics. In: Proceedings of Workshop on GRAph Data Management Experiences and Systems, GRADES 2014, pp. 10:1–10:6. ACM, New York (2014)

The Case for Personalized Anonymization of Database Query Results

Axel Michel[1,2](✉), Benjamin Nguyen[1,2], and Philippe Pucheral[2]

[1] LIFO, INSA-CVL, Boulevard Lahitolle, Bourges, France
{axel.michel,benjamin.nguyen}@insa-cvl.fr
[2] PETRUS Team, INRIA Saclay & DAVID, UVSQ, Versailles, France
philippe.pucheral@inria.fr

Abstract. The benefit of performing Big data computations over individual's microdata is manifold, in the medical, energy or transportation fields to cite only a few, and this interest is growing with the emergence of smart disclosure initiatives around the world. However, these computations often expose microdata to privacy leakages, explaining the reluctance of individuals to participate in studies despite the privacy guarantees promised by statistical institutes.

In this paper, we consolidate our previous results to show how it is possible to push personalized privacy guarantees in the processing of database queries. By doing so, individuals can disclose different amounts of information (*i.e.* data at different levels of accuracy) depending on their own perception of the risk, and we discuss the different possible semantics of such models.

Moreover, we propose a decentralized computing infrastructure based on secure hardware enforcing these personalized privacy guarantees all along the query execution process. A complete performance analysis and implementation of our solution show the effectiveness of the approach to tackle generic large scale database queries.

1 Introduction

In many scientific fields, ranging from medicine to sociology, computing statistics on (often personal) private and sensitive information is central to the discipline's methodology. With the advent of the Web, and the massive databases that compose it, statistics and machine learning have become "data science": their goal is to turn large volumes of information linked to a specific individual, called *microdata*, into knowledge. Big Data computation over microdata is of obvious use to the community: medical data is used to improve the knowledge of diseases, and find cures: energy consumption is monitored in smart grids to optimize energy production and resources management. In these applications, real knowledge emerges from the analysis of aggregated microdata, not from the microdata itself[1].

[1] We thus do not consider applications such as targeted advertising, who seek to characterize the users at an individual level.

J. Filipe et al. (Eds.): DATA 2017, CCIS 814, pp. 261–285, 2018.
https://doi.org/10.1007/978-3-319-94809-6_13

Smart disclosure initiatives, pushed by legislators (*e.g.* EU General Data Protection Regulation [1]) and industry-led consortiums (*e.g.* blue button and green button in the US[2], Midata[3] in the UK, MesInfos[4] in France), hold the promise of a *deluge* of microdata of great interest for analysts. Indeed, smart disclosure enables individuals to retrieve their personal data from companies or administrations that collected them. Current regulations carefully restrict the uses of this data to protect individual's privacy. However, once the data is anonymized, its processing is far less restricted. This is good news, since in most cases, these operations (*i.e.* global database queries) can provide results of tunable quality when run on anonymized data.

Unfortunately, the way microdata is anonymized and processed today is far from being satisfactory. Let us consider how a national statistical study is managed, *e.g.* computing the average salary per geographic region. Such a study is usually divided into 3 phases: (1) the statistical institute (assumed to be a *trusted third party*) broadcasts a query to collect raw microdata along with *anonymity guarantees* (*i.e.*, a privacy parameter like k in the *k–anonymity* or ϵ in the *differential privacy* sanitization models) to all users; (2) each user consenting to participate transmits her microdata to the institute; (3) the institute computes the aggregate query, while respecting the announced anonymity constraint.

This approach has two important drawbacks:

1. The anonymity guarantee is defined by the querier (*i.e.* the statistical institute), and applied uniformly to all participants. If the querier decides to provide little privacy protection (*e.g.* a small k in the *k–anonymity* model), it is likely that many users will not want to participate in the query. On the contrary, if the querier decides to provide a high level of privacy protection, many users will be willing to participate, but the quality of the results will drop. Indeed, higher privacy protection is always obtained to the detriment of the quality and thus utility of the sanitized data.
2. The querier is assumed to be trusted. Although this could be a realistic assumption in the case of a national statistics institute, this means it is impossible to outsource the computation of the query. Moreover, microdata centralization exacerbates the risk of privacy leakage due to piracy (Yahoo and Apple recent hack attacks are emblematic of the weakness of cyber defenses[5]), scrutinization and opaque business practices. This erodes individuals trust in central servers, thereby reducing the proportion of citizen consenting to participate in such studies, some of them unfortunately of great societal interest.

The objective of this paper is to tackle these two issues by reestablishing *user empowerment*, a principle called by all recent legislations protecting the management of personal data [1]. Roughly speaking, user empowerment means

[2] https://www.healthit.gov/patients-families/.
[3] https://www.gov.uk/government/news/.
[4] http://mesinfos.fing.org/.
[5] Yahoo 'state' hackers stole data from 500 million users - BBC News. www.bbc.co.uk/news/world-us-canada-37447016.

that the individual must keep the control of her data and of its disclosure in any situation. More precisely, this paper, which is an extended version of [2] makes the following contributions (new contributions in bold):

– propose a query paradigm incorporating personalized privacy guarantees, so that each user can trade her participation in the query for a privacy protection matching her personal perception of the risk,
– **present and discuss possible semantics of personalized privacy queries**,
– **propose efficient algorithms to manage the collection and distributed computation of queries over large quantities of data**,
– provide a secure decentralized computing framework guaranteeing that the individual keeps her data in her hands and that the query issuer never gets cleartext raw microdata and sees only a sanitized aggregated query result matching all personalized privacy guarantees,
– conduct a performance evaluation **and large scale implementation** on a real dataset demonstrating the effectiveness and scalability of the approach.

The rest of the paper is organized as follows. Section 2 presents related works and background materials allowing the precisely state the problem addressed. Section 3 details how to manage personalized queries in sql. Section 4 covers a discussion of the semantics of this model. Section 5 discusses the complex algorithmic issues that arise due to the use of personalized anonymity. Section 6 shows the results of our implementation, demonstrating the efficiency of the approach in real case scenarios. Finally, Sect. 7 concludes.

2 State of the Art and Problem Statement

2.1 Related Works on Privacy-Preserving Data Publishing

Anonymization has been a hot topic in data publication since the 1960's for all statistical institutions wanting to publish aggregate data. The objective of most of these data publishing techniques is to provide security against an attacker who is going to mount *deanonymization* attacks, which will link some sensitive information (such as their salary or medical diagnosis) to a specific individual. Particular attention was drawn to the problem by Sweeney, the introduction of the k-anonymity model [4] that we consider in this paper. k-anonymity is a partition based approach to anonymization, meaning that the original dataset, composed of individual's microdata, is partitioned, through generalization or suppression of values, into groups who have similar values which will then be used for grouping.

The partition-based approach splits the attributes of the dataset in two categories: a quasi-identifier and some sensitive data. A quasi-identifier (denoted QID) is a set of attributes for which some records may exhibit a combination of unique values in the dataset, and consequently be identifying for the corresponding individuals (e.g., ZipCode, BirthDate, Gender). The sensitive part (denoted

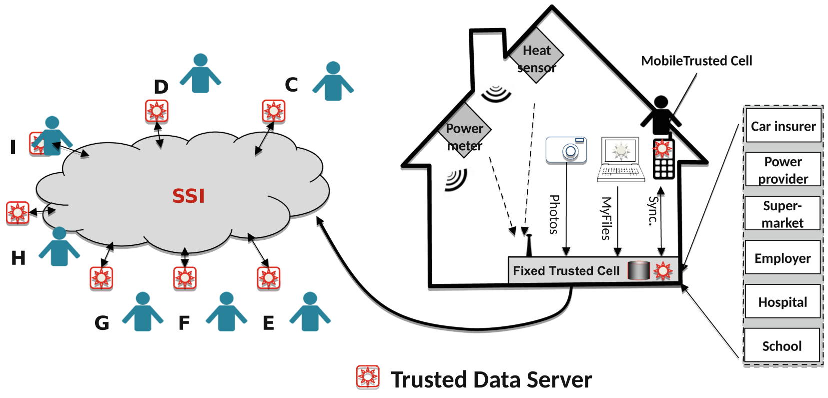


Fig. 1. Trusted Cells reference architecture [3].

SD) encompasses the attribute(s) whose association with individuals must be made ambiguous (e.g., Disease).

Partition-based approaches essentially apply a controlled degradation to the association between individuals (represented in the dataset by their quasi-identifier(s)) and their sensitive attribute(s). The initial dataset is deterministically partitioned into groups of records (classes), where quasi-identifier and sensitive values satisfy a chosen partition-based privacy model. The original k-Anonymity model [4] requires each class to contain at least k indistinguishable records, thus each sensitive data will be associated with at least k records. Many other models have been introduced since 2006, such as ℓ-Diversity [5] or t-Closeness [6]. Each model further constrains the distribution of sensitive data within each class, tackling different adversarial assumptions. For example, the ℓ-Diversity principle requires that the set of sensitive data associated to each equivalence class be linked to ℓ different sensitive values. t-closeness requires each class to have a similar distribution of sensitive values. To illustrate this, Table 1 shows a 3-anonymous and 2-diverse version of a dataset. This means, for each tuple, at least two others have the same quasi-identifier (*i.e.* 3-anonymity) and for each group of tuples with the same quasi-identifier, there are at least two distinct sensitive values (*i.e.* 2-diversity). It is important to note that the higher the k and ℓ, the better the privacy protection, but the lower the precision (or quality) of the query.

A different concept is *differential privacy*, introduced by Dwork in [7]. Differential privacy is more adapted to interactive query answering. Its' advantage is to provide formal guarantees regardless of the knowledge of the adversary. However, differential privacy limits the type of computation which can be made on the data. Moreover, fixing the privacy parameter ϵ is a cumbersome and not intuitive task, out of reach of lambda individuals.

Another approach is to make an agreement between the user and the querier. The concept of *sticky policies* presented by Trablesi et al. [8] consists in making

Table 1. 3–anonymous and 2–diverse table [2].

Quasi-identifier		Sensitive
ZIP	Age	Condition
112**	>25	Cancer
112**	>25	Cancer
112**	>25	Heart disease
1125*	*	Heart disease
1125*	*	Viral infection
1125*	*	Cancer

a policy about authorization (*i.e.* what the querier can do) and obligation (*i.e.* what the querier must do) which will stick to the user data.

The concept of personalized privacy was first introduced by Gedik and Liu [9]. They proposed an algorithm to provide location based services with privacy guarantees in a mobile network. Each users can specify the privacy parameter which a third party will ensure when querying the location service. Their algorithm is based on the *Clique* problem and called the Clique-Cloak algorithm. A graph is created linking *neighbour* users together (*i.e.* when they emit a query) and when a clique is identified, the server sends the clique to the location service. The result received by the location service is then sent to each mobile which filters the result to get the information respectively expected by each user. Other works [10, 11] follow the same principle and propose different algorithms providing personalized privacy associated to location based services. Mokbel et al. proposed the *Casper* [10] algorithm. The idea is to keep up to date location data in a trusted third party server called anonymization server. This anonymization server maintains multiple copies of the data at different levels of granularity forming a pyramid with fine grained data at the base and coarse grained data at the top. When a user queries a location based service, the anonymization server sends back the appropriate cell ensuring the k–anonymity guarantee chosen by the user. The result is finally filtered by the user mobile or the anonymization server. Bamba Bhuvan et al. introduced the principle of *PrivacyGrid* [11]. The PrivacyGrid algorithm partitions location map into rectangular cells forming a grid. Users locations are kept up to date on the grid and, when a user queries the location service, the user cell is merged with neighbour cells to satisfy the expected k parameter. These works are all based on a trusted third party which ensures the privacy guarantees. Compared to these works, the approach presented in this paper proposes a solution providing personalized privacy not dedicated to location data and without the use of a trusted (centralized) third party. Our approach is based on pre-anonymizing - any form of - data at their source, that is directly at collection time, following the user's privacy recommandations. The queries on the collected dataset are then computed in a decentralized way without the intervention of any trusted (centralized) third party. Participants in this decentralized secure protocol further group data of various individuals to match the privacy guarantees announced by the querier. Note that contrary to location based services, computing real-time queries is not the concern here.

Personalized differential privacy has been first presented by Jorgensen et al. in [12]. They proposed two mechanisms where each user specifies a personalized ϵ and where the result satisfies all users constraints. The first mechanism is naive and applies differential privacy with the greatest ϵ constraint. Their second mechanism is based on a non-uniform sampling algorithm using two parameters. The first parameter is an ϵ given by each user ensuring a personalized differential privacy and the second is a global ϵ ensuring a minimum degree of privacy. Another personalized differential privacy algorithm has been presented by Li et al. in [13]. They use a partitioning-based mechanism to achieve personalized differential privacy. Their algorithm consists in partitioning the dataset by grouping user data having close ϵ constraints and then applying an algorithm which ensures the differential privacy with the lowest ϵ value of the partition (*i.e.* the highest privacy guarantee).

Another approach is to give the possibility to the user to personalize the generalization of her sensitive attributes. Xiaokui Xiao and Yufei Tao introduce in their article [14] the concept of guarding node. Each user can specify a guarding node (*i.e.* a node on the taxonomy tree of the sensitive attribute) which limits the precision of the sensitive value. This guarantees to users that they cannot be linked to any sensitive value below the guarding node in the taxonomy tree with a given probability. So the sensitive attribute can be generalized instead of generalizing more than enough quasi-identifiers. This provides a better quality to the overall anonymized data with a guaranteed probability of identification. In this paper, we also exploit the idea of giving a greater importance to the sensitive attributes.

2.2 Reference Computing Architecture

Concurrently with smart disclosure initiatives, the *Personal Information Management System* (PIMS) paradigm has been conceptualized [15], and emerges in the commercial sphere (*e.g.* Cozy Cloud, OwnCloud, SeaFile). PIMS holds the promise of a Privacy-by-Design storage and computing platform where each individual can gather her complete digital environment in one place and share it with applications and other users under her control. The *Trusted Cells architecture* presented in [3], and pictured in Fig. 1, precisely answers the PIMS requirements by preventing data leaks during computations on personal data. Hence, we consider Trusted Cells as a reference computing architecture in this paper.

Trusted Cells is a decentralized architecture by nature managing computations on microdata through the collaboration of two parties. The first party is a (potentially large) set of personal *Trusted Data Servers* (TDSs) allowing each individual to manage her data with tangible elements of trust. Indeed, TDSs incorporate tamper resistant hardware (*e.g.* smartcard, secure chip, secure USB token) securing the data and code against attackers and users' misusages.

Despite the diversity of existing tamper-resistant devices, a TDS can be abstracted by (1) a Trusted Execution Environment and (2) a (potentially untrusted but cryptographically protected) mass storage area where the personal data resides. The important assumption is that the TDS code is executed by the secure device hosting it and then cannot be tampered, even by the TDS holder herself.

By construction, secure hardware exhibit limited storage and computing resources and TDSs inherit these restrictions. Moreover, they are not necessarily always connected since their owners can disconnect them at will. A second party, called hereafter *Supporting Server Infrastructure* (SSI), is thus required to manage the communications between TDSs, run the distributed query protocol and store the intermediate results produced by this protocol. Because SSI is implemented on regular server(s), *e.g.* in the Cloud, it exhibits the same low level of trustworthiness.

The resulting computing architecture is said *asymmetric* in the sense that it is composed of a very large number of low power, weakly connected but highly secure TDSs and of a powerful, highly available but untrusted SSI.

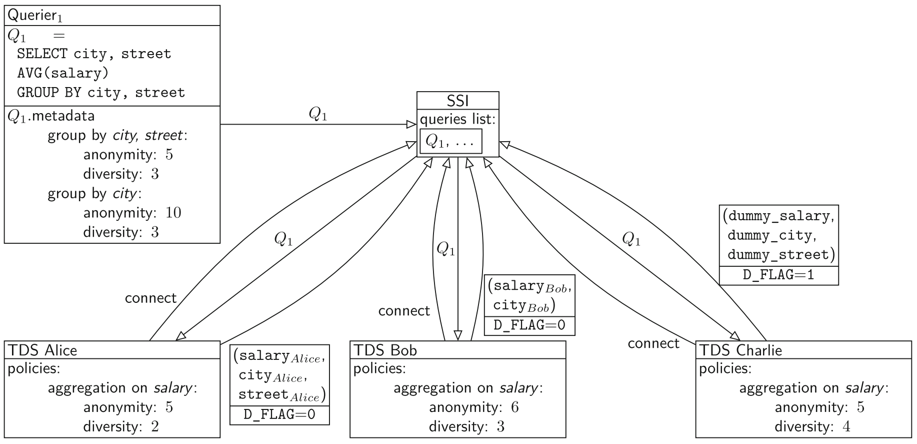


Fig. 2. Example of collection phase with anonymity constraints [2].

2.3 Reference Query Processing Protocol

SQL/AA (SQL Asymmetric Architecture) is a protocol to execute standard SQL queries on the Trusted Cells architecture [16,17]. It has been precisely designed to tackle this issue, that is executing global queries on a set of TDSs without recentralizing microdata and without leaking any information.

The protocol works as follows. Once an SQL query is issued by a querier (*e.g.* a statistic institute), it is computed in three phases: first the *collection phase* where the querier broadcasts the query to all TDSs, TDSs decide to participate or not in the computation (they send dummy tuples in that case to hide their denial of participation), evaluate the `WHERE` clause and each TDS returns its own encrypted data to the SSI. Second, the *aggregation phase*, where SSI forms partitions of encrypted tuples, sends them back to TDSs and each TDS participating to this phase decrypts the input partition, removes dummy tuples and computes the aggregation function (*e.g.* `AVG`, `COUNT`). Finally the *filtering phase*, where TDSs produce the final result by filtering out the `HAVING` clause and send the result to the querier. Note that the TDSs participating to each phase can be different. Indeed, TDSs contributing to the collection phase act as data producers while TDSs participating to the aggregation and filtering phases act as trusted computing nodes. The tamper resistance of TDSs is the key in this protocol since a given TDS belonging to individual i_1 is likely to decrypt and aggregate tuples issued by TDSs of other individuals $i_2, \ldots, i_n$. Finally, note that the aggregation phase is recursive and runs until all tuples belonging to a same group have been actually aggregated. We refer the interested reader to [2,16,17] for a more detailed presentation of the SQL/AA protocol.

2.4 Problem Statement

Hence, the problem addressed in this paper is to propose a (SQL) query paradigm incorporating personalized (k-anonymity and ℓ-diversity) privacy guarantees and enforcing these individual guarantees all along the query processing without any possible leakage.

3 Personalized Anonymity Guarantees in SQL

3.1 Modeling Anonymisation Using SQL

We make the assumption that each individual owns a local database hosted in her personal TDS and that these local databases conform to a common schema which can be easily queried in SQL. For example, power meter data (resp., GPS traces, healthcare records, etc.) can be stored in one (or several) table(s) whose schema is defined by the national distribution company (resp., an insurance company consortium, the Ministry of Health, etc.). Based on this assumption, the querier (i.e., the statistical institute) can issue regular SQL queries as shown by the Fig. 3.

```
SELECT <Aggregate function(s)>
FROM <Table(s)>
WHERE <condition(s)>
GROUP BY <grouping attribute(s)>
HAVING <grouping condition(s)>
```

Fig. 3. Regular SQL query form [2].

For the sake of simplicity, we do not consider joins between data stored in different TDSs but internal joins which can be executed locally by each TDS are supported. We refer to [16,17] for a deeper discussion on this aspect which is not central to our work in this paper.

Anonymity Guarantees are defined by the querier, and correspond to the k and ℓ values that will be achieved by the end of the process, for each group produced. They correspond to the commitment of the querier towards any query participant. Different k and ℓ values can be associated to different granularity of grouping. In the example pictured in Fig. 2, the querier commits to provide $k \geq 5$ and $\ell \geq 3$ at a (`City`, `Street`) grouping granularity and $k \geq 10$ and $\ell \geq 3$ at a (`City`) grouping granularity.

Anonymity Constraints are defined by the users, and correspond to the values they are willing to accept in order to participate in the query. Back to the example of Fig. 2, Alice's privacy policy stipulates a minimal anonymization of $k \geq 5$ and $\ell \geq 3$ when attribute Salary is queried.

According to the anonymity guarantees and constraints, the query computing protocol is as follows. The querier broadcasts to all potential participants the query to be computed along with metadata encoding the associated anonymity guarantees. The TDS of each participant compares this guarantees with the individual's anonymity constraints. This principle shares some similarities with *P3P*[6] with the matching between anonymity guarantees and constraints securely performed by the TDS. If the guarantees exceed the individual's constraints, the TDS participates to the query by providing real data at the finest grouping granularity. Otherwise, if the TDS finds a grouping granularity with anonymity guarantees matching her constraints, it will participate, but by providing a degraded version of the data, to that coarser level of granularity (looking at Fig. 2, answering the `group by city, street` clause is not acceptable for Bob, but answering just with `city` is). Finally, if no match can be found, the TDS produces fake data (called *dummy tuples* in the protocol) to hide its denial of participation. Fake data is required to avoid the querier from inferring information about the individual's privacy policy or about her membership to the `WHERE` clause of the query.

Figure 2 illustrates this behavior. By comparing the querier anonymity guarantees with their respective constraints, the TDSs of Alice, Bob and Charlie respectively participate with fine granularity values (Alice), with coarse granularity values (Bob), with dummy tuples (Charlie).

The working group *ODRL*[7] is looking at some issues similar to the expression of privacy policies. However, this paper is not discussing about how users can express their privacy policy in a standard way.

[6] https://www.w3.org/P3P/.
[7] https://www.w3.org/community/odrl/.

3.2 The ^{k_i}SQL/AA Protocol

We now describe our new protocol, that we call ^{k_i}SQL/AA to show that it takes into account many different k values of the i different individuals. ^{k_i}SQL/AA is an extension of the SQL/AA protocol [16,17] where the enforcement of the anonymity guarantees have been pushed in the *collection*, *aggregation* and *filtering* phases.

Collection Phase. After TDSs download the query, they compare the anonymity guarantees announced by the querier with their own anonymity constraints. As discussed above (see Sect. 3.1) TDSs send real data at the finest grouping granularity compliant with their anonymity constraints or send a dummy tuple if no anonymity constraint can be satisfied.

Aggregation Phase. To ensure that the anonymization guarantees can be verified at the filtering phase, clauses `COUNT(*)` and `COUNT(DISTINCT A)` are computed in addition to the aggregation asked by the querier. `COUNT(*)` will be used to check that the k–anonymity guarantee is met while `COUNT(DISTINCT A)`[8] are computed in addition to the aggregation asked by the querier. As explained next, these clauses are used respectively to check the k–anonymity and ℓ–diversity guarantees). If tuples with varying grouping granularity enter this phase, they are aggregated separately, *i.e.* one group per grouping granularity.

Filtering Phase. Besides `HAVING` predicates which can be formulated by the querier, the `HAVING` clause is used to check the anonymity guarantees. Typically, k–anonymity sums up to check `COUNT`$(*) \geq k$ while ℓ–diversity is checked by `COUNT(DISTINCT A)` $\geq \ell$. If these guarantees are not met for some tuples, they are not immediately discarded. Instead, the protocol tries to merge them with a group of coarser granularity encompassing them. Let us consider the example of Table 2(a). The tuple `(Bourges, Bv.Lahitolle, 1600)` is merged with the tuple `(Bourges, ******, 1400)` to form the tuple `(Bourges, ******, 1442.86)`. Merges stop when all guarantees are met. If, despite merges, the guarantees cannot be met, the corresponding tuples are removed form the result. Hence, the querier will receive every piece of data which satisfies the guaranties, and only these ones, as shown on Table 2(c). In this same example, note that another choice could have been done regarding tuple `(Le Chesnay, ******)`. Instead of removing it because it does not meet the privacy constraints, it could have been merged with tuple `(Le Chesnay, Dom. Voluceau)`. The result would loose in preciseness but will gain in completeness. We discuss more deeply the various semantics which can be associated to personalized anonymization in Sect. 4.

How to Generalize. The example pictured in Table 2 illustrates data generalization in a simplistic case, that is a Group By query performed on a single attribute which can be expressed with only two levels of preciseness

[8] Since this clause is an holistic function, we can compute it while the aggregation phase by adding naively each distinct value under a list or using a cardinality estimation algorithm such as *HyperLogLog* [18].

Table 2. Filtering phase [2].

(a) Example of a post aggregation phase result

city	street	AVG(salary)	COUNT(*)	COUNT (DISTINCT salary)
Le Chesnay	Dom. Voluceau	1500	6	4
Le Chesnay	******	1700	9	6
Bourges	Bv. Lahitolle	1600	3	3
Bourges	******	1400	11	7

(b) Privacy guarantees of the query

Attributes	k	ℓ
city, street	5	3
city	10	3

(c) Data sent to the querier

city	street	AVG(salary)
Le Chesnay	Dom. Voluceau	1500
Bourges	******	1442.86

`<City, Street>` and `<City>`. In the general case, Group By queries can be performed on several attributes, each having multiple levels of preciseness. To reach the same k and ℓ values on the groups, the grouping attributes can be generalized in different orders, impacting the quality of the result for the querier. For instance, if the `GroupBy` clause involves two attributes Address and Age, would it be better to generalize the tuples on Address (*e.g.* replacing `<City,Street>` by `<City>`) or on Age (replacing exact values by intervals)? The querier must indicate to the TDSs how to generalize the data to best meet her objectives, by indicating which attributes to generalize, in which order, and what privacy guarantees will be enforced after each generalization. In the following example, we consider the UCI Adult dataset [19], we define a `GroupBy` query `GB` on attributes `Age`, `Workclass`, `Education`, `Marital_status`, `Occupation`, `Race`, `Gender`, `Native_Country` and we compute the average `fnlwgt` operation

```
GB= Age, Workclass, Education, Marital_status,
    Ocupation, Race, Gender, Native_Country;
OP= AVG(fnlwgt);
MD= :                    k=5 l=3,
    age->20:             k=6 l=3,
    workclass->up:       k=8 l=4,
    education->5:        k=9 l=4,
    marital_status->up:  k=9 l=4,
    occupation->up:      k=10 l=4,
    race->up:            k=11 l=5,
    gender->del:         k=14 l=6,
    native country->del:k=15 l=7,
    age->40:             k=17 l=8;
```

Fig. 4. ^{k_i}SQL/AA query example [2].

`OP=AVG(fnlwgt)`. `MD` represents the metadata attached to the query. Each metadata indicates which k and ℓ can be guaranteed after a given generalization operation. Depending on the attribute type, generalizing an attribute may correspond to climbing up in a generalization hierarchy (for categorical attributes such as `Workclass` or `Race`) or replacing a value by an interval of greater width (for numeric values such as `Age` or `Education`). The `del` operation means that the attribute is simply removed. The ordering of the metadata in `MD` translates the querier requirements.

4 Semantic Issues Linked to K_i–Anonymisation

The example presented in Table 2 highlighted the fact that no single personalized anonymization semantics fits all situations. Typically, personalized anonymization introduces an interesting tradeoff between preciseness and completeness of the query results. Figure 5 illustrates this tradeoff and shows that two different result tables might both respect the privacy constraints, while containing different tuples. In other words, it is important to be able to define the formal characteristics that can be guaranteed, even though some are contradictory. We call *semantics* of a k_i–anonymised query the rules that lead to publishing tuples under privacy guarantees. These semantics are evaluated during the filtering phase of the query.

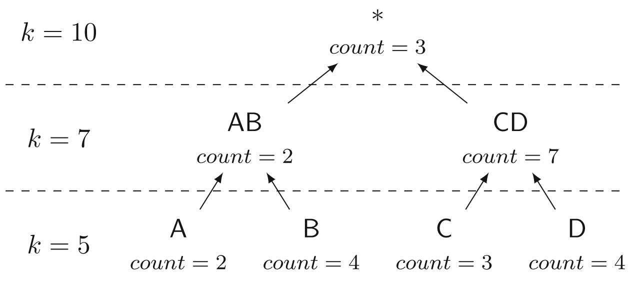


Fig. 5. Post-aggregation phase example.

Let Q be a query computing an aggregation on a set of tuples, grouping on an attribute a of finite domain $dom(a)$. We note $|dom(a)|$ the size of the domain. For the sake of simplicity, we assume that the querier specifies the generalization process by means of a labeled generalization *tree* (thus each value has only one parent) which explains how each value must be generalized, and which indicates the privacy guarantees enforced at each generalization level of a.

We plot on Fig. 5 an example where $dom(a) = \{A, B, C, D\}$. The example represents the post-aggregation result of query Q (before the filtering phase). For readability purpose, the k privacy guarantee labels are indicated on the left. Each node shows the total number of participants in the given aggregation group. It is important to note that at this point, a microdata tuple will only have contributed to a *single* group, depending on its privacy preferences.

Example. Group AB is composed of the tuples of 2 participants, with values A or B for a, and with a privacy guarantee of $k \geq 7$. Clearly as only 2 participants contributed to the group in this case, this guarantee cannot be enforced, and thus this group cannot be published *as* is in the resulting query answer.

Different decisions, which will impact the semantics of the Group By query, can be made regarding this group: (i) discard the group, and the corresponding microdata tuples, (ii) merge it with a more general group (*i.e.* a parent node in the generalization tree), (iii) add tuples from a more specific but compatible group to it. We discuss next different realistic semantics for the Group By query and how to attain them. Two concurrent objectives can actually be pursued: generalizing the microdata tuples the least possible or dropping the least possible microdata tuples. Note that when publishing the results of such queries, although a microdata tuple is only counted once, the domains of the groups can overlap.

1. ***Selective* semantics.** These semantics prioritize the precision of the result over its completeness, while enforcing all privacy guarantees. In other words, the querier wants to keep the best quality groups as possible, that is at the finest generalization level, and accepts to drop groups that do not achieve the privacy constraints.
2. ***Complete* semantics.** Conversely, these semantics prioritize the completeness of the result over its precision. The querier wants to keep the most microdata tuples as possible, and accepts to generalize and merge groups to achieve this objective.

Selective semantics are rather straightforward since they correspond to a locally optimisable algorithm. They can be implemented as the Algorithm 1.

Algorithm 1. *Selective* algorithm.

```
procedure selective(List L)
    while $N = L.pop()$ do
        if $satisfyPrivacy(N)$ then
            $publish(N)$
        else
            if $hasParent(N)$ then
                $N.mergeTo(parent(N))$
            end if
        end if
    end while
end procedure
```

With this Algorithm 1, the microdata are generalized along the generalization tree until the group cardinality reaches the privacy guarantee of a given node and are then published with this level of precision. Once a node content is published, it cannot be used by a higher level node to help it get published. Thus a node N cannot be discarded if a higher level node N' is published.

Example. Groups A and B do not respect the privacy constraint, they are thus merged with Group AB. The same holds for groups C and D which are merged with group CD. Groups AB and CD (including the newly generalized tuples) are both published. Group $*$ does not respect the privacy constraint and is discarded.

Complete semantics are more difficult to achieve. Indeed, the objective here is to globally optimize the number of tuples published, and only then look at the quality of the generalization. The problem can be formalized as a constrained optimization problem with two (ordered) objective functions. F_1 is the priority optimization function which counts the number of microdata tuple published. F_2 is the secondary optimization function which evaluates the overall quality of groups. Functions F_2 could be defined to take into account the semantic relatedness among nodes of the generalization tree by using appropriate metrics (*e.g.* Wu and Palmer [20] metric), but for the sake of simplicity we chose to define the overall quality as the $\sigma_g \in \{groups\}(depth(g) \cdot count(g))$. Maximal quality is obtained if all the microdata tuples are published at maximal depth. Dropped tuples induce maximal penalty in quality, since they do not contribute to the score. Also note that it is not possible to generalize only a subset of the microdata tuples composing one of the initial groups. A group must be generalized completely or not at all. However, descendant groups could be generalized more than some of their ancestor groups. For instance, group CD might be published as is since its already satisfies the privacy guarantee, and groups C and D might be generalized to $*$ to form a group of cardinality 10 which can also be published (instead of dropping the three initial elements in $*$). Hence, regarding F_1, various combinations are valid like $\{(*, count = 10), (AB, count = 8), (CD, count = 7)\}$, $\{(*, count = 18), (CD, count = 7)\}$ or $\{(*, count = 11), (CD, count = 14)\}$. In this example, F_2 should lead to select the former one.

The practical difficulty comes precisely from the computation of this secondary optimization function F_2 (not to mention having to chose what function to use). Indeed, otherwise a trivial answer would simply to generalize all the microdata tuples to $*$, and publish a single group. In a centralized context, a (non linear) constraint solver could be used to compute the optimal result to this problem. However, in a distributed context, generalization must take place during the distributed filtering phase. Thus we propose to use a greedy heuristic to simplify decision making in the *complete* semantics approach.

The difference between these two algorithms is that the second algorithm accepts to generalize a node which could be published as is, if it can help a higher node to get published. Also note that another semantic choice that must be made when implementing the algorithm is when deciding which descendant node to merge, when several possibilities exist.

Example. The algorithm behaves the same as the *selective* algorithm until it has groups AB and CD in R. Then it must decide whether to merge AB or

Algorithm 2. Greedy *complete* algorithm.

```
procedure $complete$(Ordered List (lower node first) L)
    $R \leftarrow$empty list
    while $N = L.pop()$ do
        if $satisfyPrivacy(N)$ then
            $R.push(N)$
        else if $hasDescendant(N)$ then
            $P \leftarrow descendant(N)$
            $R.remove(P)$
            $P.mergeTo(N)$
        else if $hasParent(N)$ then
            $N.mergeTo(parent(N))$
        end if
        $publish(R)$
    end while
end procedure
```

CD with $*$. The decision can be made by choosing the smallest group that still respects the privacy constraint. In this case, as $|AB| = 8$ and $|CD| = 14$, we would chose to merge AB with $*$ and produce $|*| = 11$ and $|CD| = 14$. However we see that the optimal solution would have been to merge C and D with $*$ or merge CD with $*$ and generalize C and D to CD, thus producing $|AB| = 8$, $|CD| = 7$ and $|*| = 10$.

Conclusion. Several different semantics can be applied in order to decide how to publish the groups while respecting privacy constraints. There is no ideal choice in absolute terms and the decision should be application driven. However, besides the preciseness and completeness of the obtained result, the cost of implementing these semantics may widely differ. While implementing the *selective* semantics is straightforward, optimizing the computation of the *complete* semantics can be rather complex and costly, especially in a decentralized context. It is part of our future work to propose and evaluate greedy algorithms tackling this challenge.

5 Algorithmic Issues Linked to K_i–Anonymisation

The implementation of ^{k_i}SQL/AA builds upon the *secure aggregation* protocol of SQL/AA [16,17], recalled in Sect. 2.3. The secure aggregation is based on non-deterministic encryption as AES CBC to encrypt tuples. Each TDS encrypts its data with the same secret key but with a different initialization vector such that two tuples with the same value have two different encrypted values. Hence, all TDSs, regardless of whether they contributed to the collection phase or not, can participate to a distributed computation and decrypt all intermediate results produced by this computation without revealing any information to the SSI. Indeed, the SSI cannot infer personal informations by the distribution of same encrypted values. Injecting personalization in this protocol has no impact on the cryptographic protection. However, this impacts the internal processing of each phase of the protocol, as detailed next.

5.1 Collection Phase

The confrontation between the querier's privacy guarantees and the TDS's privacy constraints, along with the potential data generalization resulting from this confrontation, are included in the collection phase. The resulting algorithm is given below (see Algorithm 3). As in most works related to data anonymization, we make the simplifying assumption that each individual is represented by a single tuple. Hence, the algorithm always return a single tuple. This tuple is a dummy if the privacy constraints or the `WHERE` clause cannot be satisfied.

Algorithm 3. Collection Phase.

```
procedure COLLECTION_PHASE(Query Q)
    $t \leftarrow getTuple(Q)$
    $p \leftarrow getConstraints(Q)$
    $g \leftarrow getGuarantees(Q)$
    $i \leftarrow 0$
    if $verifyWhere(t, Q)$ then
        while $g_i < p$ do
            if $not\ canGeneralize(t)$ then
                $t \leftarrow makeDummy(Q)$
            else
                $t \leftarrow nextGeneralization(t)$
                $i \leftarrow i + 1$
            end if
        end while
    else
        $t \leftarrow makeDummy(Q)$
    end if
    return $Encrypt(t)$
end procedure
```

5.2 Aggregation Phase

To make sure that aggregations are entirely computed, the SSI uses a divide and conquer partitioning to make the TDS compute partial aggregations on partitions of data. The aggregation phase, as implemented in SQL/AA [16,17], is illustrated by the Fig. 6.

The weakness of this way of computing the aggregation is the working load put on the unique TDS performing the final aggregation. The limited amount of TDSs RAM reduces the number of different groups (*i.e.* given by the `GroupBy`

$$\begin{tikzcd}
\mathsf{TDS}_1\text{: Agg}(P_1) \arrow[dr] & & \\
& \mathsf{TDS}_1\text{: Agg}(P_{1,2}) \arrow[ddr] & \\
\mathsf{TDS}_2\text{: Agg}(P_2) \arrow[ur] & & \\
& & \mathsf{TDS}_1\text{: Final Aggregation} \\
\mathsf{TDS}_3\text{: Agg}(P_3) \arrow[dr] & & \\
& \mathsf{TDS}_2\text{: Agg}(P_{3,4}) \arrow[uur] & \\
\mathsf{TDS}_4\text{: Agg}(P_4) \arrow[ur] & &
\end{tikzcd}$$

Fig. 6. Aggregation phase with four partitions.

clause) which can be managed by a single TDS. Moreover, the larger the partitions to merge, the larger the time spent by that TDS to do the computation and the greater the burden put on the TDS owner who cannot easily perform other tasks in parallel. This limitation is magnified in our context since personalized anonymization may lead a wider number of groups since initial groups may appear at different granularity levels.

We propose two algorithms overcoming this limitation. The first algorithm, called the *swap-merge* algorithm, is a direct adaptation of the SQL/AA aggregation algorithm where partitions bigger than the TDS RAM are simply swapped in NAND Flash. When it comes to computes aggregations on two sorted partitions saved on NAND Flash, the TDS will use a simple merge sort to build the result partition. The corresponding algorithm is straightforward and not detailed further due to space limitation. The benefit of this adaptation is to accomadate an unlimited amount of groups while keeping logarithmic the number of aggregation phases. However, this benefit comes at the price of ever increasing the load imposed on TDSs at the root of the aggregation tree due to NAND Flash I/Os. Note that executing the initial SQL/AA aggregation algorithm in streaming to avoid resorting to NAND swapping at each TDS would reveal the tuple ordering to the SSI. This option is then discarded.

The second algorithm, illustrated by the Fig. 7, does not rely on NAND Flash swapping and is called the *network-merge*. The idea is to use a bubble sort like algorithm to merge two sorted partitions.

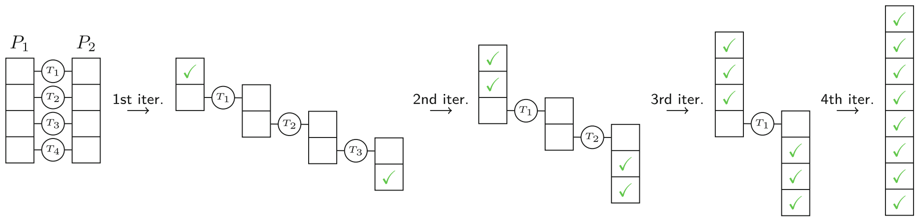


Fig. 7. Merging sorted partitions without NAND access.

The Fig. 7 illustrates the merge of two sorted partitions P_1 and P_2. We assume that each partition is in turn decomposed into n frames ($n = 4$ in the figure) in order to accommodate the RAM capacity of a TDS. More precisely, the maximal size of a frame is set to the half of the TDS RAM size (so that two frames of two different partitions can fit in RAM and be merged without swapping), minus a RAM buffer required to produce the merge result. Let P_i^j denote the j^{th} frame of partition P_i and let P_r denote the partition resulting from the merge of P_1 and P_2. Assuming that P_1 and P_2 are internally sorted in ascending order on the grouping attributes, the merge works as follows. For each $j = 1..n$, the j^{th} frames of P_1 and P_2 are pairwise sent to any merger TDS[9] which produces in return the j^{th} and $(j+1)^{th}$ sorted frames of P_r. All elements in P_r^j are smaller or equal to the elements of P_r^{j+1}. By construction, after the first iteration of the protocol, frame P_r^1 contains the smallest elements of P_r and frame P_r^n the greatest ones. Hence, these two frames do not need to participate to the next iterations. Hence, after k iterations, $2 \cdot k$ frames of P_r are totally sorted, the smallest elements being stored in frames $P_r^{1..k}$ and the greatest ones being stored in frames $P_r^{n-k..n}$. The algorithm, presented in Algorithm 4 is guaranteed to converge in a linear number of iterations. For readability concern and without loss of generality, we suppose that P_1 and P_2 have the same size.

Algorithm 4. Partition Merger Algorithm.

```
procedure NETWORK-MERGE(Partition $P_1$, Partition $P_2$, Command cmd)
    Let $top, bot, mid, q_1, q_2$ empty partitions
    for $i$ from 1 to $NumberOfFrame(P_1)$ do
        $Send(P_1^i, P_2^i, cmd)$ to $TDS_i$
    end for
    for $i$ from 1 to $NumberOfFrame(P_1)$ do
        $P_r \leftarrow receive()$ from $TDS_i$
        if $i == 1$ then
            $top.append(P_r^1)$
        else
            $q_1.append(P_r^1)$
        end if
        if $i == NumberOfFrame(P_1)$ then
            $bot.append(P_r^2)$
        else
            $q_2.append(P_r^2)$
        end if
    end for
    if $NotEmpty(q_1)$ then
        $mid \leftarrow$ NETWORK-MERGE$(q_1, q_2, cmd)$
    end if
    return $top + mid + bot$
end procedure
```

[9] Each TDS can contribute to any phase of the protocol, depending on its availability, independently of the fact that it participated to the collection phase.

Hence, algorithm *network-merge* can accommodate any number of groups without resorting to NAND Flash swapping. The price to pay is a linear (instead of logarithmic for *swap-merge* algorithm) number of iterations. The negative impact on the latency of the global protocol is low compared to the latency of the collection phase itself. But the benefit is high in terms of TDS availability. Thanks to this algorithm, each participating TDS contributes to a very small part of the global computation and this part can additionally be bounded by selecting the appropriate frame size. This property may be decisive in situations where TDSs are seldom connected.

5.3 Filtering Phase

The filtering phase algorithm is given by Algorithm 5.

First, the algorithm sorts tuples of the aggregation phase by generalization level, making multiple sets of tuples of same generalization level. The function `verifyHaving` checks if `COUNT(*)` and `COUNT(Disctinct)` match the anonymization guarantees expected at this generalization level. If so, the tuple is added to the result. Otherwise, it is further generalized and merged with the set of higher generalization level. At the end, every tuple which cannot reach the adequate privacy constraints, despite achieving maximum generalization, is not included in the result.

Algorithm 5. Filtering Phase.

```
procedure FILTERING_PHASE(Query Q, TuplesSet T)
    $sortByGeneralizationLevel(T)$
    $g \leftarrow getGuarantees(Q)$
    for $i$ from 0 to $MaxGeneralizationLevel(Q)$ do
        for $t \in T_i$ do
            $t \leftarrow decrypt(t)$
            if $verifyHaving(t, Q)$ then
                $result.addTuple(t)$
            else if $canGeneralize(t)$ then
                $t \leftarrow nextGeneralization(t)$
                $T_{i+1}.addTuple(t)$
            end if
        end for
    end for
    return $result$
end procedure
```

6 Experimental Evaluation

We have implemented the ^{k_i}SQL/AA protocol on equivalent open-source hardware used in [16,17]. The goal of this experimental section is to show that there is very little overhead when taking into account personalized anonymity constraints compared to the performance measured by the original implementation of To et al. Our implementation is tested using the classical adult dataset of UCI-ML [19].

6.1 Experiments Platform

The performance of the ^{k_i}SQL/AA protocol presented above has been measured on the tamper resistant open-source hardware platform shown in Fig. 8. The TDS hardware platform consists of a 32-bit ARM Cortex-M4 microcontroller with a maximum frequency of 168 MHz, 1 MB of internal NOR flash memory and 196 kb of RAM, itself connected to a μSD card containing all the personal data in an encrypted form and to a secure element (open smartcard) containing cryptographic secrets and algorithms. The TDS can communicate either through USB (our case in this study) or Bluetooth. Finally, the TDS embeds a relational DBMS engine, named PlugDB[10], running in the microcontroller. PlugDB is capable to execute SQL statement over the local personal data stored in the TDS. In our context, it is mainly used to implement the WHERE clause during the Collection phase of ^{k_i}SQL/AA.

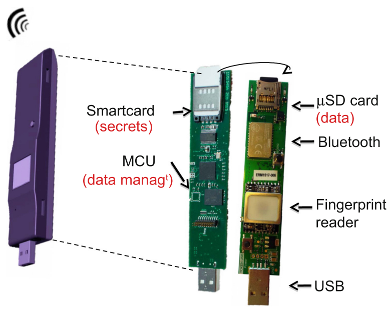


Fig. 8. TDS characteristics [21].

Our performance tests use the Adult dataset from the UCI machine learning repository [19]. This dataset is an extraction from the American census bureau database. We modified the dataset the same way of [22,23]. We kept eight attributes to perform the `GoupBy` clause, namely `age`, `workclass`, `education`, `marital status`, `occupation`, `race`, `native country` and `gender`. Since our work is based on GROUP BY queries, we also kept the `fnlwgt` (*i.e.* final weight) attribute to perform an `AVG` on it. The final weight is a computed attribute giving similar value for people with similar demographic characteristics. We also removed each tuple with a missing value. At the end we kept 30162 tuples.

[10] https://project.inria.fr/plugdb/en/.

Attributes `age` and `education` are treated as numeric values and others as categorical values. Since TDS have limited resources, categorical value are represented by a bit vector. For instance, the categorical attribute `workclass` is represented by a 8 bits value and its generalization process is performed by taking the upper node on the generalization tree given in Fig. 9. The native country attribute is the largest and requires 49 bits to be represented.

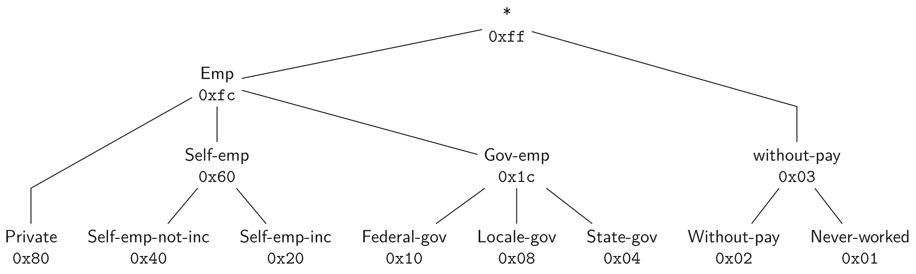


Fig. 9. Generalization tree of `workclass` attribute [2].

6.2 Performance Measurements

^{k_i}SQL/AA being an extension of SQL/AA protocol, this section focuses on the evaluation of the overhead incurred by the introduction of anonymization guarantees in the query protocol. Then, it sums up to a direct comparison between ^{k_i}SQL/AA and the genuine SQL/AA. To make the performance evaluation more complete, we first recall from [17] the comparison between SQL/AA itself and other state of the art methods to securely compute aggregate SQL queries. This comparison is pictured in Fig. 10. The Paillier curve shows the performance to compute aggregation in a secure centralized server using homomorphic encryption, presented in [24]. The DES curve uses also a centralized server and a DES encryption scheme (data are decrypted at computation time). Finally, SC curves correspond to the SQL/AA computation with various numbers of groups G (*i.e.* defined by `GroupBy` clause). This figure shows the strength of the massively

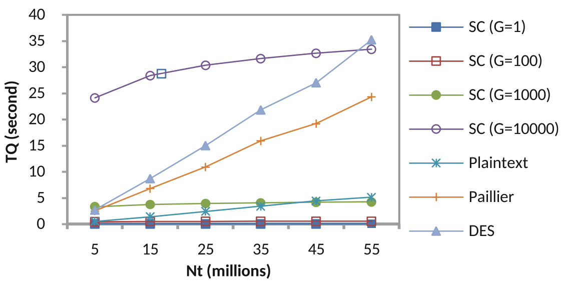


Fig. 10. Performance measurements of SQL/AA and state of the art [17].

parallel calculation of TDSs when G is small and its limits when G is really too big. We compare next the overhead introduced by our contribution to SQL/AA.

Categorical vs. Numeric Values. We ran a query with one hundred generalization levels, using first categorical, then numerical values. Execution time was exactly the same, demonstrating that the cost of generalizing categorical or numerical values is indifferent.

Collection and Filtering Phases. Figures 11 and 12 show the overhead introduced by our approach, respectively on the collection and filtering phases. The time corresponds to processing every possible generalization of the query presented in Fig. 4, which generates the maximal overhead (i.e., worst case) for our approach. The *SQL/AA* bar corresponds to the execution cost of the SQL/AA protocol inside the TDS, the *data transfer* bar corresponds to the total time spent sending the query and retrieving the tuple (approximately 200 bytes at an experimentally measured data transfer rate of 7.9 Mbits/s), the *TDS platform* bar corresponds to the internal cost of communicating between the infrastructure and the TDS (data transfer excluded), and the *Privacy* bar corresponds to the overhead introduced by the ^{k_i}SQL/AA approach. All times are indicated in milliseconds. Values are averaged over the whole dataset (i.e. 30K tuples).

Collection Phase Analysis. The overhead of the collection phase resides in deciding how to generalize the tuple in order to comply with the local privacy requirements, and the global query privacy constraints. Figure 11 shows that our protocol introduces about a low overhead (between 0.29 ms and 0.40 ms). The variation is due to the number of generalization to be made. Most of the time is taken by the DBMS running in the TDS since it takes privacy preference on the NAND memory which is slow. The time taken by the SQL/AA system is almost due to the query to get data on the NAND memory which is *constant*. The same query could have been kept but we used the minimal query size to show this variation. This performances are a bit different from the article [2] since we used more complicated queries on the TDS DBMS. Our contribution introduces an overhead under 10% of the overall time which we consider as a very reasonable cost.

Filtering Phases Analysis. Figure 12 shows breakdown of the filtering phase execution time. The filtering phase takes place once the TDSs have computed all the aggregations and generalizations. The limited resources of the TDSs are

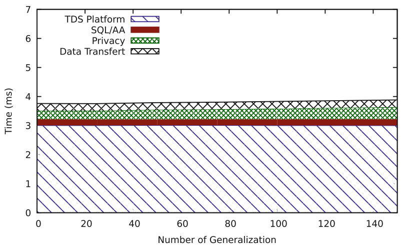


Fig. 11. Collection phase execution time breakdown.

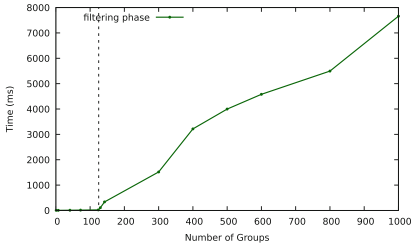


Fig. 12. Filtering phase execution time breakdown.

bypassed by the SQL/AA system with the help of the (distributed) aggregation phase. Since every group is represented by one tuple, the TDS which computes the filtering phase receives a reduced amount of tuples (called G). To et al. have shown that the SQL/AA protocol converges if it is possible for a given TDS to compute G groups during the aggregation phase. As this is the number of tuples that will be processed during the filtering phase, we know that if G is under the threshold to allow its computation via the distributed aggregation phase, then it will be possible to compute the filtering phase with our improved protocol. To make performance measures, the cortex-M4 has a 24 bit system timer giving the possibility to measure time lower than 100 ms. Since computations now use NAND access, the time is too high to give a comparison between the SQL/AA system and the ^{k_i}SQL/AA system. The dashed line shows the limit where resorting to NAND is mandatory.

Aggregation Phase Analysis. Figure 13 shows the performance measures of the sorting algorithm used in the aggregation phase. After the aggregation phase, the result contains as much tuples as different groups made by the `GroupBy` clause. We vary the number of groups to increase or decrease the overall time taken by each step of the merge algorithm. The limit of the SQL/AA system is shown by the dashed vertical line. The system was unable to compute a query with more groups than this limit. The merge algorithm permits to overcome this limit. The performance measure was done on one TDS. Since each step of the merge sort can be distributed, the maximum time of each partition merge was kept and summed between the different steps to get the time showed by the Fig. 13. The performance of the swap merge drastically decreases when the data to aggregate cannot fit in the RAM of the TDS and needs to be swapped on the NAND memory.

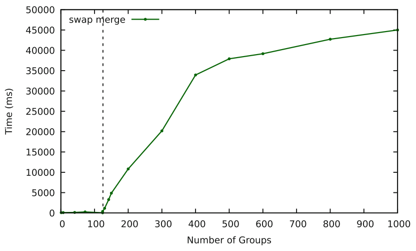


Fig. 13. Aggregation phase execution time breakdown.

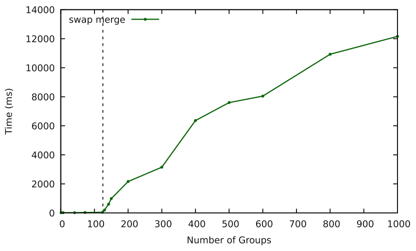


Fig. 14. TDS availability required by the aggregation phase.

One of the weakness of the swap merge algorithm is the time a TDS must be available to merge two partitions. This time is directly related to the number of distinct groups generated by the `GroupBy` clause. It is important to measure this time. The maximum time a TDS must be available to compute the aggregation phase is illustrated by the Fig. 14. The time increases drastically when the TDS needs to use the NAND memory.

7 Conclusion

While privacy-by-design and user empowerment are being consecrated in the legislations regulating the use of personal data (e.g., EU General Data Protection Regulation), time has come to put these principles into practice. This paper tackles this objective by proposing a novel approach to define personalized anonymity constraints on database queries. The benefit is twofold: let the individuals control how their personal data is exposed ever since collection time and, by this way, provide the querier with a greater set of consenting participants and more accurate results for their surveys. Moreover, we propose a fully decentralized and secure execution protocol enforcing these privacy constraints, which avoids the risk of centralizing all personal data on a single site to perform anonymization. Our experiments shows the practicality of the approach in terms of performance. Finally, this paper shows that different semantics can be attached to the personalized anonymization principle, opening exciting research perspectives for the data management community.

References

1. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance)
2. Michel, A., Nguyen, B., Pucheral, P.: Managing distributed queries under personalized anonymity constraints. In: Sixth International Conference on Data Science, Technology and Applications, DATA 2017 (2017)
3. Anciaux, N., Bonnet, P., Bouganim, L., Nguyen, B., Popa, I.S., Pucheral, P.: Trusted cells: a sea change for personal data services. In: Online Proceedings of Sixth Biennial Conference on Innovative Data Systems Research, CIDR 2013, Asilomar, CA, USA, 6–9 January 2013 (2013)
4. Sweeney, L.: k-anonymity: a model for protecting privacy. Int. J. Uncertain. Fuzziness Knowl. Based Syst. **10**, 557–570 (2002)
5. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: l-diversity: privacy beyond k-anonymity. In: Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, Atlanta, GA, USA, 3–8 April 2006, p. 24 (2006)
6. Li, N., Li, T., Venkatasubramanian, S.: Closeness: a new privacy measure for data publishing. IEEE Trans. Knowl. Data Eng. **22**, 943–956 (2010)
7. Dwork, C.: Differential privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 1–12. Springer, Heidelberg (2006). https://doi.org/10.1007/11787006_1
8. Trabelsi, S., Neven, G., Raggett, D., Ardagna, C., et al.: Report on design and implementation. Technical report, PrimeLife Deliverable (2011)
9. Gedik, B., Liu, L.: Location privacy in mobile systems: a personalized anonymization model. In: 25th IEEE International Conference on Distributed Computing Systems (ICDCS 2005), pp. 620–629 (2005)
10. Mokbel, M.F., Chow, C.Y., Aref, W.G.: The new casper: query processing for location services without compromising privacy. In: Proceedings of the 32nd International Conference on Very Large Data Bases, VLDB 2006, pp. 763–774. VLDB Endowment (2006)

11. Bamba, B., Liu, L., Pesti, P., Wang, T.: Supporting anonymous location queries in mobile environments with privacygrid. In: Proceedings of the 17th International Conference on World Wide Web, WWW 2008, pp. 237–246. ACM, New York (2008)
12. Jorgensen, Z., Yu, T., Cormode, G.: Conservative or liberal? Personalized differential privacy. In: 2015 IEEE 31st International Conference on Data Engineering, pp. 1023–1034 (2015)
13. Li, H., Xiong, L., Ji, Z., Jiang, X.: Partitioning-based mechanisms under personalized differential privacy. In: Kim, J., Shim, K., Cao, L., Lee, J.-G., Lin, X., Moon, Y.-S. (eds.) PAKDD 2017. LNCS (LNAI), vol. 10234, pp. 615–627. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57454-7_48
14. Xiao, X., Tao, Y.: Personalized privacy preservation. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, SIGMOD 2006, pp. 229–240. ACM, New York (2006)
15. Abiteboul, S., André, B., Kaplan, D.: Managing your digital life. Commun. ACM **58**, 32–35 (2015)
16. To, Q., Nguyen, B., Pucheral, P.: SQL/AA: executing SQL on an asymmetric architecture. PVLDB **7**, 1625–1628 (2014)
17. To, Q.C., Nguyen, B., Pucheral, P.: Private and scalable execution of SQL aggregates on a secure decentralized architecture. ACM Trans. Database Syst. **41**, 16:1–16:43 (2016)
18. Flajolet, P., Fusy, É., Gandouet, O., Meunier, F.: Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm. In: Proceedings of the 2007 International conference on Analysis of Algorithms (AOFA 2007) (2007)
19. Lichman, M.: UCI machine learning repository (2013)
20. Wu, Z., Palmer, M.: Verbs semantics and lexical selection. In: Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics, ACL 1994, pp. 133–138. Association for Computational Linguistics, Stroudsburg (1994)
21. Lallali, S., Anciaux, N., Sandu Popa, I., Pucheral, P.: A secure search engine for the personal cloud. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, SIGMOD 2015, pp. 1445–1450. ACM, New York (2015)
22. Iyengar, V.S.: Transforming data to satisfy privacy constraints. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2002, pp. 279–288. ACM, New York (2002)
23. Bayardo, R.J., Agrawal, R.: Data privacy through optimal k-anonymization. In: Proceedings of the 21st International Conference on Data Engineering, ICDE 2005, pp. 217–228. IEEE Computer Society, Washington, DC (2005)
24. Ge, T., Zdonik, S.: Answering aggregation queries in a secure system model. In: Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB 2007, pp. 519–530. VLDB Endowment (2007)

Author Index